TECHNIQUES ALGORITHMIQUES POUR LA FIABILITÉ ET LA SÛRETÉ DES SYSTÈMES DISTRIBUÉS

par

Ayman Farhat

mémoire présenté au Département de mathématiques et d'informatique
en vue de l'obtention du grade de maître ès sciences (M.Sc.)

FACULTÉ DES SCIENCES
UNIVERSITÉ DE SHERBROOKE

Sherbrooke, Québec, Canada, juin 1998

Le _____ , le jury suivant a accepté ce mémoire dans sa version finale.
date

Président-rapporteur: M. Ernest Monga _____
Département de mathématiques et d'informatique

Membre: M.Shengrui Wang _____
Département de mathématiques et d'informatique

Membre: M. Béchir Ayeb _____
Département de mathématiques et d'informatique

Sommaire

La fiabilité des systèmes est indispensable dans de nombreuses applications. À cet effet, la tolérance aux fautes est une approche largement utilisée dans les systèmes qui requièrent une fiabilité élevée. Son but est de fournir un service malgré la présence de composants fautifs dans le système en fonctionnement. Le problème des généraux Byzantins est un modèle puissant, car il n'impose aucune contrainte sur le comportement des composants fautifs. Ceux-ci peuvent se comporter de façon arbitraire.

Dans ce mémoire, nous allons tout d'abord présenter le problème des généraux Byzantins traditionnel. Ensuite, nous démontrerons que sans ajout de contraintes au modèle initial, l'identification des composants fautifs est impossible. Enfin, nous donnerons les conditions nécessaires pour qu'une telle identification puisse opérer.

Remerciements

Je remercie Béchir Elayeb, mon directeur de recherche, pour toute l'aide qu'il m'a apportée tout au long de ma maîtrise. Qu'il trouve ici l'expression de mes sentiments les meilleurs.

Mes remerciements vont aussi à toutes les autres personnes qui m'ont aidé d'une manière ou d'une autre.

Finalement, je présente ma reconnaissance à ma famille et à tous mes amis, qui ont toujours cru en mes capacités pour mener à terme ce travail.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

Introduction

Problématique

L'utilisation croissante des ordinateurs conduit à une nécessité : avoir des systèmes d'ordinateurs extrêmement fiables. Ces systèmes, qui sont souvent composés de plusieurs centaines d'ordinateurs, apparaissent comme un outil puissant et économique dans les applications industrielles, militaires et éducationnelles. Avec cette prolifération, les besoins pour de tels systèmes ont augmenté. Ces systèmes sont malheureusement composés, dans la plupart des cas, de composants non fiables dont les pannes peuvent causer des catastrophes, peut-être même une perte de vie humaine tels dans un crash d'avion, ou une catastrophe nucléaire. Il est clair que la fiabilité de ces systèmes est indispensable dans de telles applications et que les besoins en fiabilité vont continuer à croître.

Typiquement, un système comprend plusieurs composants, chacun d'eux étant en lui-même un système autonome. Le composant fautif peut être défectueux sans contaminer les autres composants du système. Toutefois, il est aussi possible qu'un composant défectueux contamine certains autres composants du système et, de ce fait, il devient une source d'erreur.

La composition d'un système à partir de plusieurs composants élémentaires offre un avantage de taille : créer un système dont la puissance globale est supérieure à celle de chacun de ses composants. Malheureusement, si un ou plusieurs composants du système sont fautifs, la puissance globale peut être affectée. Il est alors essentiel de fournir un mécanisme pour détecter et localiser, ou au moins masquer, les composants fautifs dans ces systèmes. Dans la suite, nous allons traiter la notion de panne ou de faute de système. Succinctement, un système est dit en panne si son fonctionnement observé diffère de celui qui est spécifié. En fait, la notion de composant fautif n'est pas simple à définir et il existe plusieurs modèles de fautes que nous allons décrire au chapitre 1. Toutefois, on peut dire qu'une panne de système est causée par les composants fautifs de ce système.

La tolérance aux fautes est une approche largement utilisée dans les systèmes qui requièrent une fiabilité élevée. Son but est de fournir un service malgré la présence des composants fautifs dans le système en fonctionnement. Il existe une variété de techniques utilisées pour la tolérance aux fautes [15]. Dans ce mémoire, nous nous intéressons aux techniques appliquées dans les systèmes distribués. Les techniques de tolérance aux fautes sont basées sur l'utilisation de la redondance pour masquer l'effet des fautes dans les systèmes distribués.

Succinctement, un système distribué est un réseau composé de plusieurs ordinateurs (fréquemment appelés nœuds). Ces nœuds sont géographiquement situés dans différents endroits et sont connectés. Tous les nœuds sont autonomes et communiquent les uns avec les autres via des lignes de communication. Chaque nœud est formé d'une unité qui a une mémoire locale inaccessible à partir de tous les autres nœuds et une horloge privée qui gouverne l'exécution des instructions sur ce nœud. Chaque nœud a aussi une interface par laquelle il est connecté au réseau. Les nœuds du système communiquent entre eux par des messages via les lignes de communication qui les relient.

Le problème de consensus

Le consensus est un des problèmes fondamentaux dans la conception des systèmes distribués fiables. Si la fiabilité des nœuds et des lignes de communication entre eux est assurée, la solution du problème de l'accord devient triviale : l'initiateur du message envoie simplement une copie de son message à chacun des autres nœuds. En réalité, il y a toujours une chance que la ligne de communication ou l'initiateur d'un message soit fautif. Ce sont ces fautes qui empêchent les nœuds d'atteindre un consensus.

Le problème de consensus s'occupe de l'*accord* des nœuds non fautifs sur l'état du système (sa configuration, la synchronisation de son horloge, le contenu des communications ou n'importe quelle autre valeur qui demande une consistance globale). Cet accord doit se faire malgré l'inadvertance possible ou même la propagation erronée d'informations par certains composants fautifs du système.

Les composants majeurs d'un système distribué sont les nœuds et les lignes de communication. Différentes méthodes existent pour résoudre le problème du consensus et assurer la tolérance aux fautes dans un système distribué. La plupart de ces méthodes font des hypothèses explicites mais aussi implicites sur le comportement du système et particulièrement sur le mode de pannes de ses composants [15]. Ces hypothèses sont fréquemment faites sur :

1. les relations entre les horloges des différents nœuds,
2. la disponibilité des données au-delà des pannes,
3. le comportement des nœuds durant les pannes, et
4. la fiabilité et le type du réseau.

Il a souvent été supposé que lorsqu'un composant d'un système distribué tombe en panne, il se comporte d'une façon bien définie. Voici deux exemples de ce type de pannes :

1. un composant qui donne toujours des zéros,
2. un composant qui arrête tout simplement de fonctionner.

Techniques de *SD* et de *BA*

Deux approches ont été proposées pour résoudre le problème de consensus dans un système distribué :

1. l'approche *SD* (*System Level Diagnosis*) introduite par *Preparata & al.* [17] en 1967,
2. le *BA* (*Byzantine Agreement*, nommé aussi *Byzantine Generals Problem*) mis au point par *L.Lamport, R.Shostak et M.Pease* [16], en 1980.

Plusieurs méthodes de tolérance aux fautes supposent que si un nœud tombe en panne d'une façon sympathique (bien définie), sa panne peut être détectée par les autres nœuds dans le système. Cette détection de panne exige des protocoles de diagnostic de fautes. C'est le domaine du diagnostic du système (*System Diagnosis* ou *SD*). Le *SD* consiste à diagnostiquer, c'est-à-dire, détecter et localiser les nœuds fautifs et partager cette information avec les autres nœuds non fautifs [17].

En général, lorsqu'un composant ou un système tombe en panne, son comportement peut être totalement arbitraire. Par exemple, le composant ou le système fautif peut envoyer des informations différentes aux autres composants avec lesquels il communique.

Avec ce type de pannes, atteindre l'accord entre les différents composants est un problème compliqué. Il est appelé le problème des généraux Byzantins (*Byzantine Generals Problem* ou *Byzantine agreement* ou *BA*). Les protocoles utilisés pour atteindre l'accord avec ce type de faute arbitraire seront appelés par la suite les protocoles de l'accord Byzantin (*the Byzantine agreement protocols*) [13].

Le *BA* est un problème complexe car aucune hypothèse n'est posée sur le comportement des composants fautifs. Ceux-ci peuvent se comporter de façon arbitraire. Cela est réaliste car les composants d'un système se comportent souvent arbitrairement lorsqu'ils tombent en panne.

En dépit du fait que le *BA* et le *SD* ont des caractéristiques différentes, ils poursuivent des buts similaires : atteindre l'accord entre les nœuds non fautifs malgré la présence de certains nœuds fautifs. En fait, le *SD* identifie les nœuds fautifs pour les isoler, alors que le *BA* masque les effets des nœuds fautifs [6].

Objectifs et organisation du mémoire

L'algorithme Byzantin vise à s'assurer que tous les nœuds non fautifs se mettent d'accord sur la même valeur sans identifier les fautes dans le système. Jusqu'à présent, les chercheurs ont visé uniquement le masquage des fautes sans se soucier de les identifier. Malheureusement, sans un mécanisme d'identification des composants fautifs, les algorithmes de masquage peuvent devenir désuets. En effet, les algorithmes supposent que le nombre de composants fautifs ne peut dépasser une limite fixe. Il s'ensuit alors que le mécanisme d'identification des composants fautifs est indispensable pour maintenir la validité des algorithmes de masquage. En étudiant ces algorithmes, on se rend compte

qu'il est possible d'identifier les composants fautifs, au moins dans certains cas. L'objectif de ce travail est de décrire le *BA* et d'exploiter ses algorithmes de masquage pour pouvoir identifier les composants fautifs dans le système.

Le premier chapitre est consacré à la présentation du problème des généraux Byzantins. La solution proposée dans la version originale [13] pour résoudre le *BA* est décrite. Le reste de ce chapitre est consacré à la description des principales solutions proposées pour résoudre le *BA*. Le *BA* est coûteux en terme du nombre de messages envoyés et des exigences du système (réseau connecté complet). L'objectif principal de toutes les solutions proposées est de présenter des moyens de transmission de plus en plus efficaces.

Jusqu'à présent, la plupart des travaux qui traitent le *BA* n'ont pas la capacité de détecter et a fortiori d'identifier les composants fautifs. Le second chapitre vient proposer un mécanisme qui exploite l'algorithme présenté dans [13] pour identifier le maximum de nœuds fautifs. Les limitations de ce mécanisme sont décrites.

Dans le troisième chapitre, nous proposons un algorithme qui résout le *BA* et qui identifie tous les composants fautifs dans le système. Dans ce cas, les nœuds et les lignes de communication peuvent tomber en panne. Un réseau de diffusion généralisé, efficace et pratique, est utilisé comme architecture de communication. Une comparaison entre cette architecture et les architectures classiques est aussi donnée.

CHAPITRE 1

Le problème des généraux Byzantins

1.1 Introduction

Le problème des généraux Byzantins est en fait un problème d'accord ou encore de consensus. Il a été décrit dans [13] et a été nommé originairement *"interactive consistency"* [16]. Initialement, il s'agit de synchroniser plusieurs horloges dont certaines peuvent être fautives. De plus, on suppose que le comportement d'une horloge fautive est arbitraire. Il est intéressant de voir la relation entre le nombre total d'horloges et celui des horloges fautives. En fait, si n est le nombre total d'horloges et f_h est le nombre maximal d'horloges fautives il est nécessaire que $n > 3f_h$ pour pouvoir masquer l'effet des horloges fautives.

Le terme "Généraux Byzantins" tire son origine d'une métaphore décrite dans [13]. La version originale imagine que plusieurs divisions de l'armée byzantine sont campées autour d'une ville ennemie, chaque division est commandée par son propre général. Les généraux peuvent communiquer entre eux par messages seulement. Ils doivent adopter

un plan commun de la bataille (attaquer ou non). Parmi ces généraux, un ou plusieurs peuvent être des traîtres voulant embrouiller les autres. Les généraux ont besoin d'un algorithme qui garantisse ce qui suit :

(c1) Tous les généraux loyaux adoptent le même plan de bataille. Les généraux loyaux vont appliquer le plan généré par l'algorithme, mais les traîtres sont libres de faire tout ce qu'ils veulent.

(c2) Un groupe de traîtres ne réussit pas à forcer les généraux loyaux à adopter un plan fautif. Les traîtres peuvent affecter le plan seulement si les généraux loyaux sont quasiment divisés sur le plan à suivre. Dans un tel cas, aucune décision ne peut être considérée comme fautive.

Le problème des généraux Byzantins (nommé aussi *BA*) nécessite une définition soignée. Les chercheurs qui ont étudié ce problème considèrent un système distribué composé de plusieurs nœuds qui communiquent entre eux via des lignes de communications. Les nœuds fautifs peuvent présenter des comportements arbitraires : Un nœud fautif peut envoyer différentes valeurs à différents nœuds. Le but fondamental à atteindre est d'obtenir un consensus entre tous les nœuds non fautifs. Chaque nœud doit adopter une valeur en se basant sur les valeurs reçues des autres nœuds dans le système. Compte tenu des conditions (c1) et (c2) plus haut, il faut exiger que tous les nœuds non fautifs adoptent la même valeur. Il faut aussi assurer que tous les nœuds non fautifs ont le même ensemble des valeurs. Le consensus peut alors être atteint facilement par tous les nœuds. En effet, il suffit que tous les nœuds utilisent une même procédure pour l'adoption d'une valeur.

Malheureusement, le problème est compliqué lorsqu'un nœud fautif peut envoyer différentes valeurs aux différents nœuds. Une simple méthode d'échange de messages pour la transmission des valeurs aux différents nœuds n'est pas suffisante. Plus précisement, il

est nécessaire d'assurer que tous les nœuds non fautifs adoptent une même valeur pour chaque autre nœud. Le consensus est alors immédiat. Formellement, on exige que :

(1) Tous les nœuds non fautifs adoptent une même valeur v_{n_i} pour un nœud n_i.

(2) Si le nœud n_i est non fautif, alors chaque nœud non fautif adopte la valeur envoyée par n_i.

Ces deux exigences sont appelées *interactive consistency*. Notons que si le nœud n_i est non fautif alors (1) découle de (2). Cette formulation autorise n'importe quel type de comportement durant une panne. Les nœuds peuvent se comporter d'une façon arbitraire. Il faut aussi noter qu'une solution pour ce problème doit considérer la possibilité d'un nœud fautif qui envoie délibérément des messages dans le but de contrecarrer le consensus.

Afin d'assurer les deux exigences (1) et (2), mentionnées ci-haut, pour résoudre le problème de consensus, différents algorithmes pour le *BA* ont été développés. Le but principal de ces algorithmes est d'assurer que les nœuds non fautifs finissent par atteindre l'accord général.

Ce chapitre est organisé comme suit; dans la prochaine section, les différents modèles des fautes sont passés en revue. Dans la section 1.3, nous précisons les caractéristiques d'un système distribué requis pour assurer l'existence d'une solution de *BA*. Les composants principaux de ce système sont présentés. Dans la section 1.4, nous donnons une procédure pour obtenir l'*interactive consistency* dans le cas d'un système composé de quatre nœuds dont un est fautif. Le but de cette section est de donner au lecteur une vue d'ensemble du problème. La section 1.5 décrit l'algorithme, nommé (Oral Message ou *OM*, proposé dans la version originale [13] pour résoudre le *BA*. La section 1.6 est consacrée à l'étude des principaux travaux qui ont traité le *BA* soit avec des messages signés,

soit avec d'autres modèles de réseaux (réseau de diffusion ou de diffusion généralisé), ou encore avec d'autres types d'accord (dégradable ou aléatoire).

1.2 Les modèles de fautes

Un modèle de fautes vise à définir le comportement d'un nœud fautif. Pour le *BA*, le modèle de faute est une description des limitations d'un nœud fautif. Savoir comment un élément de traitement tombe en panne, est la clé pour faire des hypothèses réalistes et créer des algorithmes capables d'identifier et de masquer les nœuds fautifs. Dans un système distribué, le seul comportement visible d'un nœud est la séquence de messages qu'il envoie.

Les fautes des nœuds peuvent être réparties en différentes classes. Ces classes des fautes sont au nombre de sept et sont données par la figure 1.1. Nous avons :

`Faute d'arrêt de fonctionnement avec alerte` (*Fail-Stop fault*) : Cette faute apparaît lorsqu'un nœud cesse de fonctionner et alerte les autres nœuds sur son état. On exige ici que les messages déjà envoyés par ce nœud ont été corrects jusque là.

`Faute d'arrêt de fonctionnement sans alerte` (*Crash fault*) : Cette faute apparaît lorsqu'un nœud s'interrompt et ne reprend plus son fonctionnement sans alerter les autres nœuds sur son état. Un exemple de cette classe est un nœud fautif qui s'abstient d'envoyer des messages après une période de temps arbitraire. On exige ici que les messages déjà envoyés par ce nœud ont été corrects jusque là.

`Faute d'omission` (*Omission fault*) : Cette faute apparaît lorsqu'un nœud n'arrive pas à respecter un échéancier, à commencer une tâche, ou à omettre d'envoyer des

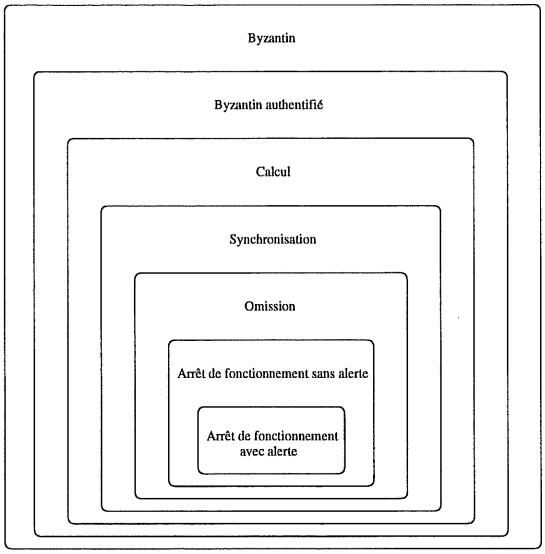


Figure 1.1 – *Les classes de fautes réparties de la plus forte à la plus faible*

messages prescrits par son protocole. On exige ici que les messages déjà envoyés par ce nœud sont corrects.

`Faute de synchronisation` (*Timing fault*) : Cette faute apparaît lorsqu'un nœud complète une tâche soit avant ou après le moment prévu, ou ne la complète pas du tout. Cette faute est parfois appelée : faute de performance.

`Faute de calcul` (*Incorrect computation fault*) : Cette faute apparaît lorsqu'un nœud échoue à produire le résultat correct en réponse à une entrée correcte.

`Faute byzantine authentifiée` (*Authenticated Byzantine fault*) : Cette faute apparaît lorsqu'un nœud envoie différents messages pendant une diffusion à ses voisins. On exige toutefois ici qu'un nœud ne peut altérer un message authentifié.

`Faute byzantine ou faute méchante` (*Byzantine fault*) : Cette faute peut être considérée comme étant l'ensemble de toutes les fautes. Un nœud fautif présente un

comportement totalement arbitraire et imprévisible. Le pire est qu'un nœud fautif soit habile pour contrecarrer la performance des nœuds non fautifs. Dans ce travail, nous allons considérer le modèle de faute byzantine, désormais appelé faute méchante.

1.3 Description du système

Dans cette section, nous précisons les caractéristiques d'un système distribué qui garantissent l'existence d'une solution de *BA*. Nous présentons aussi ses composants principaux. En première sous-section, nous discutons la synchronisation. Ensuite, dans la deuxième sous-section nous présentons les principaux composants du système : les nœuds et le mécanisme de transmission.

1.3.1 Caractéristiques du système

Le système peut être synchrone [10, 12, 13] ou asynchrone [3, 4, 8, 14]. La synchronisation permet aux nœuds non fautifs de :

- reconnaître les messages prématurés, et
- détecter les nœuds fautifs.

Sans une forme de synchronisation (c'est-à-dire qu'il n'y a pas de limite sur la vitesse relative des nœuds ou le délai des communications), le consensus avec le modèle de fautes méchantes est impossible. Ceci même si juste un seul nœud peut tomber en panne et même si le modèle de faute est la faute d'arrêt de fonctionnement sans alerte (*Crash fault*) qui est beaucoup plus bénigne que la faute méchante [7].

En fait, sans hypothèses sur la limite maximale de la durée de transmission d'un message $\triangle$ et sur la limite maximale des taux relatifs des nœuds ϕ, un nœud fonctionnant dans le protocole de consensus, peut simplement s'interrompre et retarder la procédure indéfiniment. En effet, si $\triangle$ ou ϕ est illimité, alors le consensus est impossible même dans le cas d'une seule faute. Il est évident qu'aucun algorithme ne peut résoudre le *BA* si les messages peuvent être transmis arbitrairement. On doit cependant noter une exception fondée sur l'accord aléatoire. Cette exception, proposée dans [3], sera exposé à la section 1.6.3.

1.3.2 Composants du système

Les nœuds

Tout nœud peut être fautif ou non. Chaque nœud a une mémoire locale et les nœuds non fautifs maintiennent leurs propres horloges qui ne doivent pas différer de plus d'une constante connue.

Les nœuds peuvent fonctionner dans un environnement synchrone où les nœuds envoient et reçoivent des messages en des temps bien définis. Ils peuvent également fonctionner en asynchrone. Les nœuds peuvent alors attendre une période de temps arbitraire, mais finie avant d'envoyer ou de recevoir un message.

Les mécanismes de transmission

Les nœuds vont communiquer par des messages via le mécanisme de transmission. Le contenu de ces messages est complètement sous le contrôle de l'émetteur. Il est supposé que chaque nœud exécute des algorithmes qui nécessitent l'émission des messages pour

d'autres nœuds. Les principaux mécanismes de transmission sont :

- réseau connecté complet (*Fully Connected Network*) : c'est un réseau point à point, c'est-à-dire il existe une ligne de communication entre chaque paire de nœuds. Un nœud peut chaque fois envoyer un message pour au plus un autre nœud.

- réseau de diffusion (*BroadCast network*) : un nœud peut chaque fois diffuser un message pour tous les autres nœuds (*One to All*).

- réseau de diffusion généralisé (*Generalized Network*) : les nœuds sont organisés en groupes. Cette architecture est fondée sur les réseaux de diffusion. Ce modèle combine les deux modèles précédents. Il sera exposé dans le chapitre 3.

1.4 Procédure pour atteindre l'*interactive consistency* dans le cas d'une seule faute

Pour donner aux lecteurs une première vue du problème, nous donnons une procédure pour atteindre les deux exigences de *BA* (*"interactive consistency"*) dans le cas d'une seule faute [16]. Le nombre total de nœuds est égal à quatre dont un est fautif c'est-à-dire $n = 4$ et $f_n = 1$. La procédure consiste en un échange de messages, suivi par une vérification pour montrer que les deux exigences de *BA* sont atteintes. Cette vérification est basée sur le résultat de l'échange. La figure 1.2 décrit la topologie du système considéré.

Supposons que chaque nœud veuille partager sa valeur secrète avec les autres nœuds. Deux tours d'échange d'information sont requis pour atteindre l'accord. Dans le premier

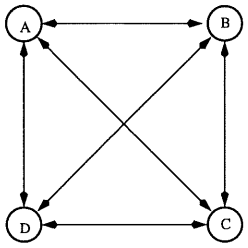


Figure 1.2 – *Quatre nœuds reliés par des lignes de communication bidirectionnelles et privées*

tour, les nœuds échangent leurs propres valeurs, par exemple :

- B envoie des messages aux nœuds A, C et D, en leur disant sa valeur.

Dans le second tour, les nœuds échangent toutes leurs informations obtenues dans le premier tour, par exemple :
B envoie un message à :

- A en lui disant les valeurs qu'il a reçues de C et D.
- C en lui disant les valeurs qu'il a reçues de A et D.
- D en lui disant les valeurs qu'il a reçues de A et C.

Un nœud fautif (s'il y en a un) peut "mentir" (envoyer une valeur différente que la valeur reçue ou tout simplement refuser d'envoyer des messages). Si un nœud non fautif ne réussit pas à recevoir un message prévu, alors il assignera simplement une valeur prédéfinie (la valeur de défaut) à ce message.

Dès que l'échange est terminé, chaque nœud doit adopter une seule valeur associée à chacun des autres nœuds. Cette valeur est obtenue en examinant les trois rapports reçus

de cette valeur. En effet, chaque nœud a trois rapports pour la valeur associée à chaque autre nœud, par exemple,

- B a reçu des valeurs de A, directement de A et via C et D.

Si au moins deux rapports parmi les trois s'accordent, la valeur majoritaire sera utilisée, c'est-à-dire la valeur qui se répète le plus souvent, sinon une valeur par défaut sera utilisée.

Vérifions que cette procédure en deux tours réponde aux deux exigences de *BA*. La première exigence qui impose que tous les nœuds non fautifs adoptent la même valeur pour la source. La deuxième impose que si la source est non fautive, alors chaque nœud non fautif adopte sa valeur.

On va traiter le cas d'un A fautif et d'un A non fautif et on va vérifier si B, C et D adoptent la même valeur pour A.

- Si A est non fautif alors B, C et D recevront chacun au moins deux rapports corrects de la valeur de A : soit de A lui même et soit d'au moins un nœud non fautif parmi les deux autres nœuds. Ainsi, la valeur majoritaire et la valeur de A s'accordent. Cette valeur majoritaire sera adoptée par tous les nœuds non fautifs et Les deux exigences sont atteintes.

- Si A est fautif, alors B, C et D devront adopter une même valeur (soit une valeur envoyée par A, soit la valeur de défaut). Si A s'abstient d'envoyer des messages, alors tous les nœuds non fautifs adoptent la valeur de défaut et les deux exigences sont atteintes. Si un nœud non fautif adopte une valeur v pour A, autre que la valeur de défaut, alors pour atteindre les deux exigences il faudra que les deux autres nœuds non fautifs adoptent cette même valeur v. Disons qu'un nœud non

fautif B, sans perte de généralité, adopte une valeur v envoyée par A, alors B a certainement reçu la même valeur v d'une combinaison de A et de deux autres nœuds non fautif, C et D.

- Si les rapports de v, reçus par B, proviennent de C et D, alors chacun de ces deux nœuds (C et D) ont reçu v de chaque nœud autre que B (et peut être de B aussi). Il s'ensuit que C et D adoptent v et les deux exigences sont atteintes.
- Dans l'autre cas, les rapports de v reçus par B, proviennent de A et d'un autre nœud non fautif, disons C sans perte de généralité. Donc, D recevra v de B et de C et il adopte v. B et C ont reçu v de chaque nœud autre que D, donc ils adoptent v. Les deux exigences sont encore atteintes.

1.5 L'algorithme de consensus OM

Nous discuterons de l'algorithme qui a été nommé (Oral Message ou OM), proposé dans la version originale [13] pour résoudre le problème de Généraux Byzantins dans un système distribué. Ce système est synchrone. Il est composé de n nœuds dont au plus f_n nœuds peuvent être fautifs. Le réseau garantit un temps limité de livraison de messages pour tous les nœuds. Les nœuds peuvent communiquer directement entre eux par les échanges de messages dans un réseau connecté complet. Chaque nœud possède une horloge, ainsi les nœuds non fautifs maintiennent un temps local qui ne diffère pas par plus d'une constante connue.

1.5.1 Nombre de nœuds fautifs permis par OM

Nous allons voir qu'il n'existe pas une solution utilisant les messages oraux qui peut tolérer un seul nœud fautif dans un système composé de trois nœuds. Soit un système

composé de trois nœuds dont un est fautif. Supposons que les nœuds ont à se mettre d'accord sur une valeur booléene : vrai(1) ou faux(0). Considérons les deux scénarios de la figure 1.3.

Figure 1.3 – *Deux scénarios différents d'un système composé de trois nœuds dont un est fautif.*

Dans le premier scénario, le nœud n_j est fautif, et bien que la source s lui ait envoyé la valeur 1, il transmet au nœud n_i la valeur 0. Dans le deuxième scénario, la source est fautive, elle a envoyé 1 au nœud n_i et 0 au nœud n_j et ce dernier l'envoie loyalement à n_i. Ces deux situations sont indiscernables pour le nœud n_i, car il ne peut pas savoir laquelle des valeurs envoyées est correcte. Celle de la source ou celle de n_j.

Le *BA* exige que si la source est non fautive, alors tous les nœuds non fautifs doivent adopter sa valeur. Dans le premier scénario, n_i ne peut pas adopter la valeur 0, sinon L'exigence du *BA*, susmentionné, ne sera pas respecté. Supposons que le nœud n_i décide d'adopter toujours dans des situations similaires la valeur de la source. Dans le deuxième scénario, n_i se met dans une situation similaire au premier scénario (n_i a reçu la valeur 1 de s et la valeur 0 de n_j). Son comportement sera le même dans ces deux scénarios : adopter la valeur envoyée par la source. Le nœud n_i adopte la valeur 1, alors que dans le même scénario n_j adopte aussi la valeur envoyée par la source (c'est-à-dire 0). Ceci ne respecte pas l'exigence de *BA* : tous les nœuds non fautifs doivent adopter la même valeur.

Il a été démontré dans [13, 16] qu'avec des messages oraux, il est impossible de résoudre

ce problème à moins que plus des deux tiers des nœuds soient non fautifs. D'une façon générale, si n est le nombre total de nœuds et f_n est le nombre de nœuds fautifs maximal, l'exigence théorique pour résoudre le BA est $n > 3f_n$.

1.5.2 Nombre de tours et de messages requis par OM

On a montré qu'il n'existe aucun algorithme déterministe pour l'accord Byzantin qui peut atteindre l'accord dans moins que $f_n + 1$ tours. Le nombre de tours de $OM(f_n)$ est ainsi optimal. De plus, le nombre des messages requis par $OM(f_n)$ est exponentiel, puisque chaque invocation de $OM(f_n - i)$ entraîne à $n-i-1$ invocations de $OM(f_n - i - 1)$, faisant ainsi croître le nombre de messages proportionnellement à $(n-1)(n-2)...(n-f_n-1)$. Sachant que f_n peut être $(n-1)/3$ nous obtenons la complexité des messages exponentiels $O(n^{f_n})$.

1.5.3 Hypothèses

L'algorithme OM exige les hypothèses suivantes :

1. L'envoyeur d'un message est toujours identifiable par le récepteur.

2. Les lignes de communication sont supposées sûres.

3. Le nombre de nœuds fautifs est $f_n < n/3$.

Remarquons que nous n'avons fait aucune hypothèse sur le comportement des nœuds fautifs. Ceci n'est pas étonnant puisque nous adoptons le modèle de faute méchante.

1.5.4 L'algorithme OM

Cet algorithme *travaille par tours*. Chaque tour consiste en des échanges de messages entre les nœuds. Dans le premier tour, la source envoie des valeurs pour les $n-1$ autres nœuds. Le nœud receveur ne peut pas avoir confiance dans la valeur qu'il a reçue de la source, puisque la source peut être fautive. Par conséquent, un nœud receveur doit savoir toutes les valeurs envoyées par la source aux autres nœuds et ainsi déterminer la valeur qui se répète le plus souvent. Cette valeur sera considérée par ce nœud comme la valeur envoyée par la source.

En d'autres termes, dans le $2^{\text{ème}}$ tour chaque receveur non fautif doit envoyer aux autres nœuds la valeur qu'il a reçue de la source dans le premier tour. Ce $2^{\text{ème}}$ tour fait que chacun des receveurs agit comme une source et envoie des messages à tous les nœuds autre que la source et (lui-même). Nous pouvons le considérer comme un système de $n-1$ nœuds, dans lequel chaque nœud envoie $n-2$ messages. Le but essentiel des messages envoyés par un nœud n_i dans ce $2^{\text{ème}}$ tour est celui d'informer les autres nœuds de la valeur qu'il a reçue de la source.

Cependant, un receveur ne peut avoir confiance dans la source dans ce second tour puisqu'un nœud qui envoie la valeur de la source originale peut être fautif. Il s'ensuit qu'un troisième tour est nécessaire. Dans ce $3^{\text{ème}}$ tour, le système peut être considéré comme étant formé de $n-2$ nœuds, et chaque nœud enverra, pour chaque message qu'il a reçu dans le $2^{\text{ème}}$ tour, sa version pour les $n-3$ nœuds restant. En raison de l'existence des nœuds fautifs, un receveur ne peut pas avoir confiance dans les sources de ce tour. Alors un $4^{\text{ème}}$ tour est requis, et ainsi de suite.

Finalement au tour $f_n + 1$, un nœud transmettra pour chaque message qu'il a reçu

dans le tour f_n, sa version pour les $n - f_n$ nœuds restants. Ainsi, la chaîne de la récursion est achevée. Chaque nœud va adopter la majorité des valeurs reçues. Cette adoption est fondée sur une fonction *majorité*. Cette fonction prend comme entrée un ensemble des valeurs, soit $\{v_i, 1 \leq i \leq n\}$. Elle retourne la valeur majoritaire[1] parmi les v_i si elle existe, si non la valeur de défaut. Pour illustrer le concept de valeur majoritaire, considérons l'ensemble $\{v_1, v_1, v_2, v_3, v_4\}$, la valeur majoritaire de cet ensemble est v_1. De même, la valeur majoritaire de l'ensemble $\{v_1, v_1, v_2, v_2\}$ n'existe pas et la valeur de défaut sera prise.

L'algorithme proposé pour résoudre le BA comme décrit dans [13].

1. La valeur qui se répète le plus souvent

$OM(f_n)$:

si $(f_n = 0)$ *alors*

(1) La source envoie sa valeur à chacun des nœuds restants.

(2) Chaque nœud utilise la valeur qu'il a reçue de la source ou la valeur de défaut s'il n'a rien reçu.

sinon

(1) La source envoie sa valeur à chaque nœud.

(2) Pour chaque i, soit v_i la valeur que le nœud n_i a reçu de la source ou la valeur de défaut s'il n'a rien reçu. Le nœud n_i agit comme source dans l'algorithme $OM(f_n - 1)$ pour envoyer la valeur v_i à chacun des $n - 2$ nœuds restants.

(3) Pour chaque nœud n_i et chaque $j \neq i$ soit v_j la valeur que le nœud n_i a reçu du nœud n_j dans (2) ou la valeur de défaut s'il n'a rien reçu. Le nœud n_i utilise la majorité de $(v_1, ..., v_{n-1})$.

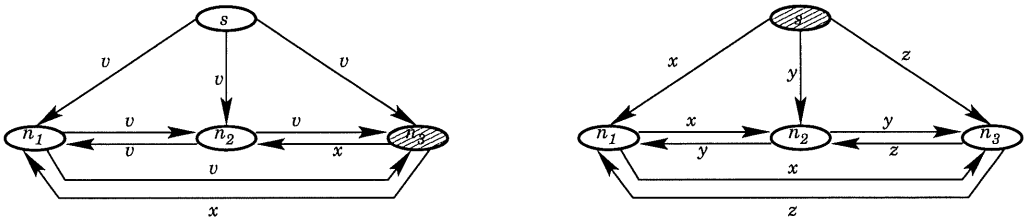
TAB. 1.1 – *L'algorithme OM.*

1.5.5 Illustration

Dans un premier temps, nous allons illustrer l'algorithme OM sur un système avec quatre nœuds dont un est fautif. Dans un $2^{\text{ème}}$ temps, nous allons illustrer la trace de l'échange de messages requis par l'algorithme OM dans un système de sept nœuds dont deux sont fautifs.

Cas de quatre nœuds dont un est fautif

Pour mieux comprendre le protocole, considérons un système composé de quatre nœuds (la source s et les trois nœuds n_1, n_2 et n_3) avec un nœud fautif. Dans un premier temps, supposons que le nœud n_3 est fautif (figure 1.4.(a)). Dans ce cas, la source va envoyer la même valeur, soit v, pour les trois autres nœuds (dont un parmi eux est fautif). Ensuite, les deux nœuds non fautifs vont loyalement envoyer la valeur qu'ils ont reçue de la source v aux autres nœuds. Le nœud fautif peut envoyer n'importe quelle valeur aux autres nœuds. Supposons qu'il envoie x pour les deux nœuds n_1 et n_2. Dans ce cas, le nœud n_1 et le nœud n_2 vont recevoir les valeurs (v, v, x). La majorité appliqué à (v, v, x) est v. Ainsi, la valeur adoptée par le nœud n_1 et le nœud n_2 est la même que la valeur envoyée par la source non fautive.



(a) Senario 1: le noeud n_3 est fautif

(b) Senario 2: la source s est fautive

Figure 1.4 – *Messages échangés dans l'algorithme* $OM(1)$

Dans un deuxième temps, supposons que la source s est fautive et les receveurs sont non fautifs (figure 1.4.(b)). Supposons que la source fautive envoie x pour le nœud n_1, y pour le nœud n_2 et z pour le nœud n_3. Puisque les nœuds n_1, n_2 et n_3 sont non fautifs, ils vont loyalement transmettre aux autres la valeur qu'ils ont reçue de la source s. Chaque nœud reçoit ainsi les valeurs (x, y, z). Chacun exécute le même algorithme de *majorité*. Ils adoptent donc la valeur majoritaire si elle existe, sinon la valeur de défaut.

Trace des messages échangés entre les nœuds

Dans cet exemple, nous allons illustrer la trace des messages échangés entre les nœuds. Considérons une source s qui envoie un message à six nœuds. Appelons les nœuds n_a, n_b, n_c, n_d, n_e et n_f et notons les messages a, b, c, d, e et f. On pourrait mettre cela sous forme d'arbre où chaque lien indique où l'information est passé. La racine est évidemment la source. En parcourant l'arbre de la racine à une feuille on a ainsi le cheminement d'un message. La figure 1.5 illustre notre tâche.

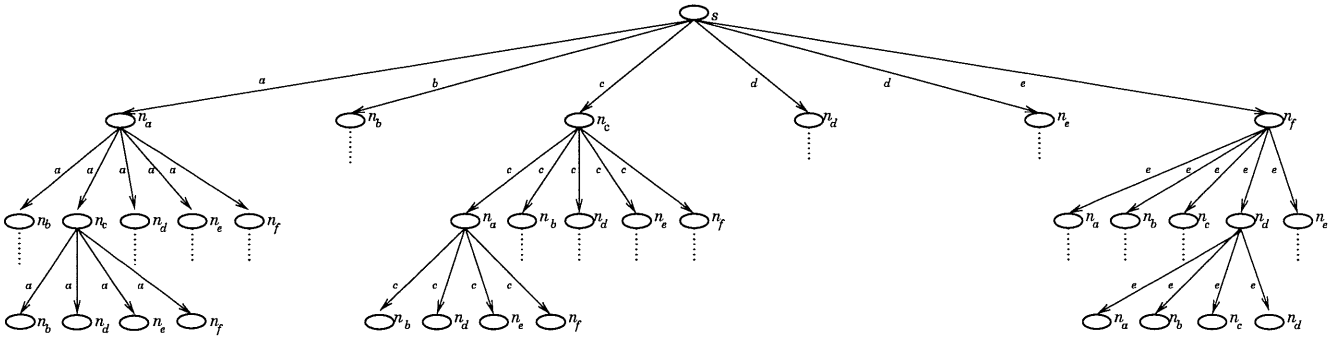


Figure 1.5 – *Cheminement de messages*

Le $2^{\text{ème}}$ niveau indique les messages reçus de la source. C'est le premier tour d'échange de messages. Dans le $3^{\text{ème}}$ niveau, nous voyons que chaque nœud a envoyé la valeur qu'il a reçue de la source dans le premier tour à chacun des autres nœuds. Par exemple, le nœud n_a envoie la valeur a à n_b, n_c, n_d, n_e et n_f. C'est le second tour d'échange de messages. Dans le troisième tour, chaque nœud envoie la valeur qu'il a reçue dans le second tour à tous les autres nœuds. Par exemple, n_c envoie la valeur reçue de n_a à n_b, n_d, n_e et n_f.

1.6 Autres travaux sur le BA

Le BA est un problème fondamental et plusieurs variantes de ce problème ont été étudiées. Traditionnellement, le BA a été étudié dans un réseau connecté complet avec l'hypothèse que seuls les nœuds peuvent être fautifs. Le BA a été réexaminé avec des messages signés [13], dans d'autres types d'accord (dégradable ou aléatoire) [9, 11, 20] et dans différents types de réseaux (réseau connecté incomplet [5, 13], réseau de diffusion [1, 2], réseau de diffusion généralisé [21, 22]). Dans certains autres travaux [23, 24], le modèle des fautes méchantes a été considéré non seulement sur les nœuds mais aussi sur les lignes de communication.

L'utilisation des méthodes comme les techniques de messages signés ou ceux qui fournissent un mécanisme de diffusion, contraint les nœuds en imposant des limites sur le caractère malicieux des nœuds fautifs. Dans ce qui suit, nous allons brièvement présenter les principaux travaux qui ont traité le BA.

1.6.1 Messages signés

Le protocole $OM(f_n)$ est compliqué car un receveur d'un message n'a aucune façon pour déterminer si l'envoyeur a altéré le message original. Le problème devient plus facile si on restreint la capacité des nœuds à altérer les messages. Cela peut être atteint lorsque chaque nœud attache une signature digitale à son message. La définition d'un message signé suppose que :

1. Un message signé par un nœud non fautif ne peut pas être contrefait.

2. Toute corruption du message est détectable.

3. La signature peut être authentifiée par tout autre nœud.

Dans ce cas, aucun nœud ne peut altérer le contenu du message d'un nœud non fautif sans que l'altération ne soit détectée par les autres nœuds non fautifs. Ainsi lorsqu'un nœud non fautif envoie un message aux autres nœuds, un nœud fautif ne peut pas altérer son message et l'envoyer aux autres nœuds. S'il tente de le faire, l'altération sera détectée. Dans cette situation, le BA est résolu et un nombre arbitraire de nœuds fautifs peut être tolérés.

Évidemment, cela limite le comportement d'un nœud fautif. Le BA devient plus simple si les messages sont signés (ou authentifiés).

1.6.2 Accord dégradable

Dans [20], on a proposé un protocole d'accord qui accomplit :

1. Le BA jusqu'à m nœuds fautifs ($0 < m \leq n/3$).
2. Une forme dégradée d'accord pouvant tolérer plus de m, mais moins de u nœuds fautifs avec ($m \leq u < n$).

Cette forme dégradée d'accord permet aux nœuds non fautifs de se mettre d'accord sur au plus deux valeurs différentes, dont une est nécessairement la valeur de défaut. L'autre est la valeur de la source si elle est non fautive.

Soit f_n le nombre de nœuds fautifs, le protocole d'accord dégradable exige les conditions suivantes :

(D.1) Tous les nœuds non fautifs adoptent une même valeur v_{n_i} pour un nœud n_i.

(D.2) Si le nœud n_i est non fautif, alors chaque nœud non fautif adopte la valeur envoyée par n_i.

(D.3) Si la source est non fautive, alors les nœuds non fautifs peuvent être divisés, au plus, en deux classes :

1. les nœuds non fautifs dans une classe doivent se mettre d'accord sur la valeur de la source,
2. et les nœuds non fautifs dans l'autre classe doivent se mettre d'accord sur la valeur de défaut.

(D.4) Si la source est fautive, alors les nœuds non fautifs peuvent être divisés, au plus, en deux classes :

1. Les nœuds non fautifs dans une classe doivent se mettre d'accord sur la valeur de défaut.
2. Les nœuds non fautifs de l'autre classe doivent se mettre d'accord sur une même valeur.

Les deux conditions (D.1) et (D.2) sont identiques aux exigences du *BA*. Remarquons que :

- si $f_n \leq m$, les conditions D.1 et D.2 ci-dessus doivent être satisfaites.
- si $m < f_n \leq u$, les conditions D.3 et D.4 ci-dessus doivent être satisfaites.

Les conditions (D.3) et (D.4) définissent l'accord dégradé et sont appliqués avec plus de m, mais moins de u nœuds fautifs. Quand $m = u$, l'accord dégradable est équivalent à *BA*.

Le tableau 1.2 présente le nombre minimum de nœuds nécessaires pour différentes valeurs

de m et u.

u / m	1	2	3	4	5
1	4	5	6	7	8
2	-	7	8	9	10
3	-	-	10	11	12

TAB. 1.2 – *Nombre minimum de nœuds nécessaires pour différentes valeurs de m et u.*

1.6.3 Accord aléatoire

L'impossibilité d'avoir un protocole d'accord déterministe dans un système asynchronisé a amené à l'étude de l'accord aléatoire (randomized solution). L'idée de l'algorithme de l'accord aléatoire est que dans n'importe quel tour particulier, il y a une probabilité pour que le nœud fautif puisse contrarier le consensus. Toutefois, on suppose qu'il y a une probabilité pour que cela ne se passe pas dans un temps prévisible. En effet les algorithmes aléatoires de *BA* ont été proposés car le nombre de tours et le nombre de messages sont faibles par rapport aux algorithmes initiaux déterministes. Le système peut être asynchrone mais l'algorithme peut prendre un nombre infini de tours avant d'atteindre l'accord. En pratique, la probabilité d'un nombre infini de tours est presque nulle [3].

1.6.4 Autres types de réseaux

L'algorithme $OM(f_n)$ fonctionne dans un réseau complètement connecté. Les nœuds peuvent ainsi communiquer directement entre eux. Évidemment, le problème devient

plus compliqué si le réseau n'est pas complètement connecté. Il a été montré dans [13] qu'avec des modifications mineures à l'algorithme $OM(f_n)$, l'accord peut être accompli dans un réseau qui n'est pas complètement connecté.

Pour un réseau connecté complet avec la panne de nœuds et des lignes de communication, Yan et Chin dans [23] ont développé un algorithme optimal pour forcer chaque nœud non fautif à atteindre l'accord en utilisant $f_n + 2$ tours, où $f_n \leq [(n-1)/3]$ et le nombre total de composants fautifs (nœuds et lignes de communication) est plus petit que $[n/2] - 1$.

Dans un réseau de diffusion, Babaoglu [1] a montré qu'un seul tour est requis pour résoudre le *BA* lorsque seuls les nœuds peuvent être fautifs. Avec la panne de nœuds et de lignes de communication, un algorithme à deux tours a été proposé par Babaoglu [2] dans un réseau de diffusion.

Dans [22], un réseau de diffusion généralisé qui combine le réseau connecté complet et le réseau de diffusion connecté est proposé. Ce type de réseau sera exposé au chapitre 3.

1.7 Conclusion

Les algorithmes traitant le *BA* opèrent sur des systèmes distribués. Succinctement, un système distribué consiste en des nœuds autonomes qui sont connectés les uns aux autres par des lignes de communication. Les nœuds sont couplés deux à deux, n'ont pas de mémoire partagée, et communiquent entre eux par des messages. Le réseau est complètement connecté. Les protocoles de communication sont utilisés pour envoyer des messages d'un nœud à un autre.

Le modèle logique consiste en des nœuds et des lignes de communication entre les nœuds. Les nœuds communiquent entre eux via les lignes de communications. Un système est dit synchrone si, en absence de pannes, il accomplit toujours sa fonction dans une période de temps finie et connue. Un système synchronisé a des lignes de communication synchronisés dans lesquels le délai maximum de transmission des messages est connu et limité. Les nœuds sont aussi synchrones, si le temps pour exécuter une séquence des instructions est fini et limité.

Les composants d'un système qui peuvent tomber en panne sont les nœuds, la mémoire, l'horloge, les lignes de communication et le logiciel. Les fautes de ces composants peuvent être classifiées comme appartenant à l'une de ces catégories: *arrêt de fonctionnement avec alerte, arrêt de fonctionnement sans alerte, faute d'omission, faute de synchronisation et faute méchante.*

L'algorithme OM, décrit précédemment, travaille par tours, et il est très coûteux en messages nécessaires pour atteindre l'accord. L'algorithme $OM(f_n)$ requiert en fait un nombre exponentiel de messages. Il a été montré qu'il n'y a pas un algorithme déterministe pour l'accord Byzantin qui peut atteindre l'accord dans moins que $f_n + 1$ tours. Le comportement arbitraire d'un nœud fautif implique une défense coûteuse.

D'autres algorithmes ont été étudiés pour résoudre le BA. Toutes les solutions proposées sont déterministes et garantissent l'accord avec un certain coût en terme de tours et de messages. Nous avons vu en bref une solution de BA, avec des messages signés, qui a simplifié le problème. Une autre fondée sur l'accord dégradé et finalement une dernière qui utilise un type particulier de réseau (réseau incomplet ou réseau de diffusion).

CHAPITRE 2

Identification des nœuds fautifs pour le problème des généraux Byzantins traditionnel

2.1 Introduction

L'algorithme de consensus OM, proposé dans [13], vise que tous les nœuds non fautifs se mettent d'accord sur la même valeur en dépit de la présence de certains composants fautifs. Jusqu'à présent, la plupart des travaux traitant le BA n'ont pas la capacité de détecter et à fortiori à identifier les composants fautifs. Chaque nœud ne connaît pas ainsi les composants fautifs dans le système.

Le BA est ardu car l'information envoyée par un nœud à un autre peut être erronée. Intuitivement, lorsqu'un nœud, disons n_i, doit vérifier l'état d'un autre nœud, disons n_j, il est conduit à disposer de la valeur qu'il a reçu directement du nœud n_j. Il doit égale-

ment disposer des valeurs que le nœud n_j a envoyé aux autres nœuds. Il s'ensuit que le nœud n_i doit communiquer avec les autres nœuds afin de disposer de ces valeurs. L'étude de l'algorithme OM montre qu'il est quand même possible d'identifier les composants fautifs, au moins dans certains cas.

Un système fiable doit être capable de tolérer la présence d'un ou de plusieurs nœuds fautifs. La tolérance signifie ici que le système continue à fournir le service attendu malgré la présence des nœuds fautifs. Malheureusement, un nœud fautif présente souvent un comportement qui n'est pas facile à modéliser. Par exemple, l'envoi d'informations incohérentes d'un nœud fautif aux autres nœuds est un casse-tête lors de la modélisation. En fait, le caractère aléatoire des comportements des nœuds fautifs empêche toute tentative de modélisation.

Dans ce chapitre, nous allons présenter un mécanisme qui exploite l'algorithme OM, pour identifier le maximum des nœuds fautifs. Les objectifs et les hypothèses pour satisfaire le mécanisme d'identification des fautifs sont présentés dans la prochaine section. Dans la section 2.3, une description de notre mécanisme est interprétée et une discussion pour décrire ses limitations est présentée. Nous terminons cette section par une présentation de l'algorithme *FIBA* (*Fault Identification Byzantine Agreement*) qui satisfait le *BA* et qui identifie le maximum des nœuds fautifs. Dans la section 2.4, une solution variante avec des messages signés est proposée. La dernière section est consacrée à la description d'un exemple qui illustre l'éfficacité et les limitations de *FIBA*.

2.2 Objectifs et hypothèses

L'objectif de ce chapitre est de concevoir un algorithme qui réalise les buts suivants :

`Satisfaction de` BA :

Les nœuds non fautifs adoptent la même valeur envoyée par une source s. Notons que si la source s est non fautive, alors tous les nœuds non fautifs adoptent la valeur envoyée par s. Autrement, les nœuds non fautifs adoptent une valeur prédéfinie.

`Identification des nœuds fautifs` :

Chaque nœud non fautif doit identifier le maximum des nœuds fautifs.

Dans ce chapitre nous prenons toutes les hypothèses sur les nœuds, les communications, les ordres des messages et le réseau, élaborées lors de la résolution du BA [13]. Plus précisement, le système est synchrone. Il contient n nœuds dont au plus f_n nœuds peuvent être fautifs, sans causer une panne dans le fonctionnement du système. Les nœuds peuvent communiquer directement entre eux par les échanges de messages dans un réseau connecté complet. Chaque nœud possède une horloge; ainsi les nœuds non fautifs maintiennent un temps local qui ne diffère pas par plus d'une constante connue. L'envoyeur d'un message est toujours identifiable par le récepteur. Seuls les nœuds peuvent être fautifs. En particulier, les lignes de communication sont supposées sûres. Aucune hypothèse n'est prise sur le comportement des nœuds fautifs. Ceci n'est pas étonnant puisque nous adoptons le modèle de la faute méchante.

2.3 Mécanisme d'identification des nœuds fautifs

Nous allons présenter un mécanisme pour identifier et masquer les nœuds fautifs. Dans un système distribué, le seul comportement visible d'un nœud est la séquence de mes-

sages qu'il a envoyé. Chaque nœud non fautif est responsable d'examiner, de comparer les messages qui lui parviennent et de construire sa propre liste des nœuds fautifs. Le reste de cette section est consacré à la description de notre mécanisme. Un algorithme d'identification des nœuds fautifs dans un format similaire à l'algorithme OM sera présenté.

2.3.1 Principes

Toutes les solutions du BA à ce jour comptent sur la redondance des nœuds et les échanges de messages entre eux pour vaincre les nœuds fautifs. Avec le grand nombre de messages transmis d'un nœud à un autre on pourrait croire qu'il serait facile d'identifier les nœuds fautifs. Malheureusement, dans plusieurs cas les nœuds fautifs peuvent rester inconnus et indiscernables. Pour pallier à ce problème, on va introduire la notion de facteur d'accusation. Dans sa forme la plus simple, un facteur d'accusation est un compteur associé à un nœud pour mesurer son degré de fiabilité. Chaque nœud associe un facteur d'accusation à tout autre nœud en fonction d'un degré de fiabilité calculé.

Soient n_i et n_j deux nœuds, notons par $fct_{n_i}^{n_j}$ le facteur d'accusation de n_i associé à n_j. Il nous faut savoir comment gérer ce facteur. Pour ce faire, nous allons profiter du fonctionnement de l'algorithme de base OM. Rappelons que l'algorithme OM travaille par tours. Chaque tour consiste en des échanges de messages entre les nœuds. Une source s envoie des valeurs aux autres nœuds. Un nœud n_i receveur doit savoir toutes les valeurs envoyées par s aux autres nœuds. De là, on peut définir les actions suivantes :

`Action d'accuser` : à chaque fois que n_i reçoit d'un nœud n_j une valeur différente de celle reçue directement de s, il accuse s et n_j en incrémentant les facteurs d'accu-

sations $fct^{s}_{n_i}$ et $fct^{n_j}_{n_i}$. Cela peut être présenté autrement. Notons par :

L : la liste des nœuds qui ont envoyé la valeur de s reçue par chacun d'eux à n_i.

v_s : la valeur envoyée par la source directement à n_i.

v_{s_x} : la valeur de la source envoyée par un nœud x à n_i.

alors on a :

$$\forall x \in L, v_s \neq v_{s_x} \Longrightarrow fct^{s}_{n_i} = fct^{s}_{n_i} + 1 \text{ et } fct^{x}_{n_i} = fct^{x}_{n_i} + 1$$

`Action d'acquitter` : une fois que n_i identifie un nœud fautif, il va acquitter tous les autres nœuds qui ont été accusés à cause de ce dernier en décrementant le facteur d'accusation de n_i associés à chacun de ces nœuds. Cela peut être présenté autrement. Notons par :

$FAUTIFS_{n_i}$: l'ensemble des nœuds considérés comme fautif par le nœud n_i.

L : liste des nœuds qui ont transmis à n_i une valeur de s différente de la valeur de s reçue directement par n_i,

alors on a :

$$s \in FAUTIFS_{n_i} \Longrightarrow \forall x \in L, x \neq s, fct^{x}_{n_i} = fct^{x}_{n_i} - 1$$

Nous allons interpréter ces deux actions dans la sous-section suivante.

Notons par f_n la limite maximale des nœuds fautifs dans le système. Nous travaillons dans un système distribué composé des nœuds autonomes. Chaque nœud construit sa propre liste des fautifs. Pour cela, il est nécessaire que chaque nœud ait une limite maximale des nœuds fautifs associés à lui, soit f_{n_i}. Initialement, f_{n_i} est égale à f_n. À chaque fois qu'un nœud n_i identifie un fautif, il décremente f_{n_i} d'une unité.

2.3.2 Interprétation

Action d'accuser

Dans l'action d'accuser, à chaque fois que n_i reçoit d'un nœud n_j une valeur différente de celle reçue directement de s, il accuse s et n_j. En effet, n_i ne peut pas accuser seulement s puisque la possibilité que n_j soit fautif existe et vice-versa. En plus, il est possible que les deux nœuds soient fautifs.

Le fait d'accuser un nœud ne veut pas dire que ce dernier va être considéré comme fautif. Plusieurs règles vont être utilisées pour identifier les nœuds fautifs en se basant sur cette accusation.

Un point important à savoir, il ne faut jamais que le même couple des nœuds soit accusé ensemble plus d'une seule fois par un nœud n_i. Sinon, il est possible que l'un de ces deux nœuds soit accusé plusieurs fois à tort à cause de l'autre.

En effet, dans un système composé de sept nœuds trois tours sont requis pour atteindre l'accord. Supposons que n_1 est fautif et considérons les deux scénarios suivantes (figure 2.1) :

- Dans le tour 2, le nœud fautif n_1 envoie x à n_3 et y à n_4. Dans le tour 3, n_3 transmet la valeur de n_1, y, à n_4. La valeur de n_1 reçue par n_4 directement de n_2 est différente de la valeur de n_1 reçue par n_4 via n_3.

- Dans le tour 2, n_3 envoie z à n_1 et n_4. Dans le tour 3, le nœud fautif n_1 change la valeur de n_3 et envoie prétendument une valeur w ($w \neq z$) à n_4. La valeur de n_3 reçue par n_4 directement de n_3 est différente de la valeur de n_3 reçue par n_4 via n_1.

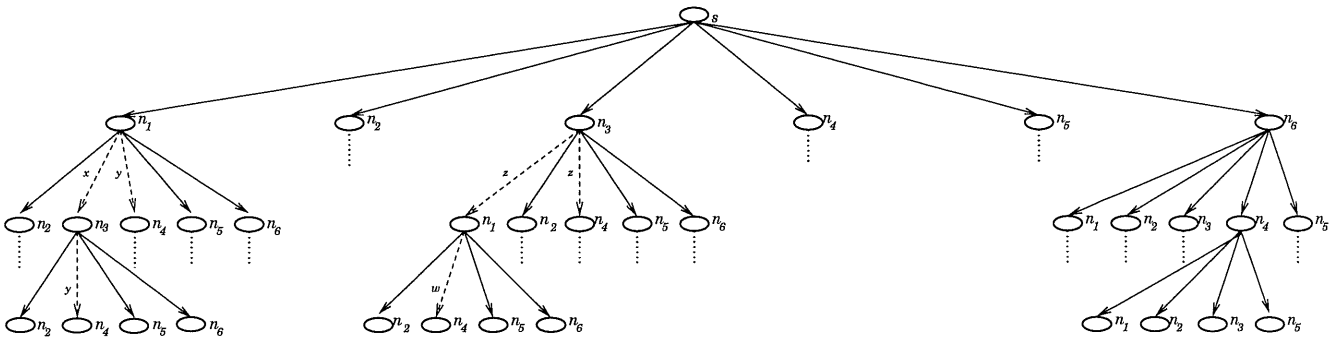


Figure 2.1 – *Trace des messages échangés*

Si l'accusation d'un même couple des nœuds est permise, le nœud n_4 accuse n_1 et n_3, dans les deux scénarios précédents, deux fois. De la même façon, il est possible que le nœud n_3 soit accusé à tort à cause d'autres nœuds fautifs. Il est évident que si un nœud n_i accuse un nœud n_j plus que f_{n_i} fois alors n_i ajoute n_j à l'ensemble des nœuds considérés comme fautif par le nœud n_i, soit $FAUTIFS_{n_i}$. Le nœud n_j peut facilement être accusé à tort plus de f_{n_i} fois. Donc, il peut être considéré comme fautif par n_i.

Action d'acquitter

L'action d'acquitter est nécessaire car sans cette action, comme nous allons voir plus loin, des nœuds non fautifs risquent d'être considérés comme fautifs. Malheureusement, il est possible qu'un ou plusieurs nœuds acquittés soient fautifs. Ceci est dû au fait qu'un nœud fautif peut contribuer à décrementer le facteur d'accusation d'un autre nœud fautif. Comme nous allons voir plus tard (section 2.3.4), la complicité des nœuds fautifs est indetectable dans les réseaux connectés complet.

Sans l'application de l'action d'acquitter, des nœuds non fautifs peuvent être considérés comme fautifs par d'autres nœuds non fautifs. En effet, considérons un système composé

de sept nœuds (s, n_1, n_2, n_3, n_4, n_5, n_6). Supposons que la source s et le nœud n_1 soient fautifs. Traitons le cas du nœud n_2 et rappelons que dans un système composé de sept nœuds f_{n_2} est égale à deux. Supposons que n_2 aie reçu 0 directement de la source s, 0 via n_1, 1 via n_3, 1 via n_4, 1 via n_5, 0 via n_6 (figure 2.2).

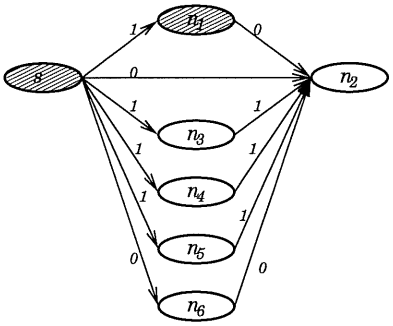


Figure 2.2 – *Valeurs reçues par n_2 directement de la source et via tous les autres nœuds.*

Le nœud n_2 accuse s et n_3, s et n_4 et s et n_5. La source s est accusée trois fois par n_2 ($3 > f_{n_2}$). Donc, n_2 identifie s comme fautive et décremente f_{n_2} de un (f_{n_2} sera 1).

À défaut d'appliquer l'action d'acquitter pour les nœuds n_3, n_4 et n_5. Dans le troisième tour d'échange, (figure 2.3), chaque nœud envoie, pour chaque valeur qu'il a reçue dans le second tour, sa version pour les cinq nœuds restants. Supposons que n_1 envoie 1 à n_2, n_4, n_5 et n_6 et 0 à n_3. Le nœud n_3 transmet loyalement à n_2 la valeur envoyée par n_1. Ce dernier accuse n_3 et n_1. Le nœud n_3 est accusé deux fois par n_2 et f_{n_2} est égal à 1, donc n_2 va considérer n_3 comme fautif.

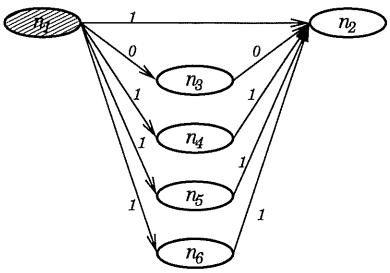


Figure 2.3 – *Valeurs reçues par n_2 directement de n_1 et via tous les autres nœuds.*

2.3.3 Règles d'identification des nœuds fautifs

Plusieurs règles peuvent être utilisées pour identifier les nœuds fautifs [6]. Chaque nœud va utiliser les règles qui lui conviennent car chaque nœud reçoit une série de messages différents. Nous allons donner deux règles, nommées *règle de zèle fort* et *règle de zèle faible*, qui serviront à identifier les nœuds fautifs.

Les nœuds fautifs peuvent être identifiés si l'une des deux règles suivantes s'applique :

Soient n_i et n_j deux nœuds, f_{n_i} la limite maximale des nœuds fautifs associés à n_i et $fct_{n_i}^{n_j}$ le facteur d'accusation de n_i associé à n_j,

`règle de zèle fort` : si $fct_{n_i}^{n_j} > f_{n_i}$ alors le nœud n_i ajoute le nœud n_j à $FAUTIFS_{n_i}$.

`règle de zèle faible` : soit $\mathcal{A}$ l'ensemble des nœuds n_j tel que $fct_{n_i}^{n_j} > 0$. Si

- $\exists\ n_j$, tel que $fct_{n_i}^{n_j} \geq f_{n_i}$,
- $card(\mathcal{A}) > 2 * f_n$.

Alors le nœud n_i ajoute le nœud n_j à $FAUTIFS_{n_i}$.

Pour mieux comprendre la deuxième règle, considérons un système composé de $n = 13$ nœuds. $\mathcal{S} = \{n_i, 1 \leq i \leq 13\}$. Le nombre maximum des nœuds fautifs est $f_n = 4$. Traitons le cas du nœud n_1. Supposons que n_1 accuse les couples suivantes : (n_2, n_3), (n_2, n_4), (n_2, n_5), (n_2, n_6), (n_7, n_8) et (n_9, n_{10}). Remarquons que n_1 a accusé n_2 quatre fois et le nombre total des nœuds accusés par n_1 est égal à neuf : n_2, n_3, n_4, n_5, n_6, n_7, n_8, n_9 et n_{10}. La `règle de Zèle faible` s'applique sur le nœud n_2. Plus précisement :

- $fct_{n_1}^{n_2} = 4$,
- $card(\mathcal{A}) = 9 > 2 * f_n$.

Ainsi, le nœud n_1 ajoute le nœud n_2 à $FAUTIFS_{n_1}$. En effet, raisonnons par l'absurde en supposant que le nœud n_2 n'est pas fautif. L'accusation des couples (n_2, n_3), (n_2, n_4), (n_2, n_5) et (n_2, n_6) implique que n_3, n_4, n_5 et n_6 sont fautifs (n_2 n'est pas fautif). L'accusation du couple (n_7, n_8) implique que n_7 ou n_8 est fautif. Ceci donne un nombre des nœuds fautifs égal à cinq ce qui est impossible. En effet, par hypothèse nous avons que le nombre maximum des fautifs est égal à quatre ($f_n = 4$). Ainsi, n_2 est fautif.

2.3.4 Discussion

Il est évident que nous ne pouvons pas identifier les nœuds fautifs dans les cas où la règle de zèle fort ou la règle de zèle faible ne s'appliquent pas. Pour mieux voir le problème, considérons un système composé de quatre nœuds (figure 2.4).

Pendant les échanges de messages, les nœuds n_2 et n_3 reçoivent la valeur 1 de la source s et la valeur 0 du nœud n_1. Les nœuds n_2 et n_3 ne peuvent pas vérifier si la valeur 0 provient bien de la source s ou du nœud n_1. Il est possible que la source soit fautive et ait envoyé la valeur 0 au nœud n_1, ce dernier, à son tour, l'envoie loyalement aux autres

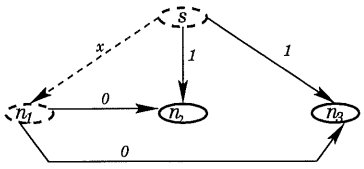


Figure 2.4 – *Valeurs envoyées par la source s et par le nœud n_1 aux nœuds n_2 et n_3 pendant les échanges de messages*

nœuds. Il est aussi possible que ce soit le nœud n_1 qui ait altéré la valeur envoyée par la source s.

Les nœuds n_2 et n_3 peuvent identifier le nœud fautif, dans le scénario précédent, s'ils utilisent des messages signés. En effet, aucun nœud ne peut altérer le contenu du message d'un nœud non fautif sans que l'altération ne soit détectée par les autres nœuds non fautifs. Ainsi lorsqu'un nœud non fautif envoie un message aux autres nœuds, un nœud fautif ne peut pas altérer son message et l'envoyer aux autres nœuds. S'il tente de le faire, l'altération sera détectée.

Remarquons qu'un nœud fautif peut contribuer à acquitter un autre nœud fautif même avec des messages signés. Les messages envoyés ne comptent pas et on ne peut rien déduire. En fait, l'un des gros problèmes d'identification est justement la complicité des nœuds fautifs. En effet, considérons un système composé de sept nœuds dont s et n_1 sont fautifs. Supposons que la source s envoie la valeur 0 à n_1 et la valeur 1 à tous les autres nœuds. Le nœud n_1 qui est fautif envoie la valeur 1 aux autres nœuds. En fait, les autres nœuds ne peuvent pas détecter l'erreur et identifier les nœuds fautifs. Malgré le fait que la source s est fautive car elle a envoyée différentes valeurs, elle ne sera pas identifiée par les autres nœuds car le nœud n_1 qui est aussi fautif altère la valeur de s. Cette altération n'est pas détectée car elle a corrigé la valeur de s. Du point de vue de tous les nœuds non

fautifs (n_2, n_3, n_4, n_5 et n_6) la source s a envoyé la valeur 1 à tous les nœuds du système et son erreur reste indetectable. Dans le chapitre suivant, on va voir qu'en utilisant un autre modèle de réseau la complicité des nœuds fautifs sera identifiable.

2.3.5 L'algorithme *FIBA* (*Fault Identification Byzantine Algorithm*)

Nous allons présenter cet algorithme dans un format similaire à l'algorithme OM. La quatrième étape est exclusive à notre algorithme.

FIBA(f_n) :

si $(f_n = 0)$ *alors*

(1) La source envoie sa valeur à chacun des nœuds restants.

(2) Chaque nœud utilise la valeur qu'il a reçue de la source ou la valeur de défaut s'il n'a rien reçu.

sinon

(1) La source envoie sa valeur à chaque nœud.

(2) Pour chaque nœud n_i,
Si n_i *reçoit une valeur de la source*
cette valeur est substituée à $v_{n_i s}$.
Si n_i *ne reçoit rien ou si la règle de zèle fort s'applique alors*
(a) la valeur de défaut est substituée à $v_{n_i s}$.
(b) ajouter la source à $FAUTIFS_{n_i}$.
n_i *agit comme source dans l'algorithme* $FIBA(f_n - 1)$ *pour envoyer la valeur* $v_{n_i s}$ *à chacun des* $n - 2$ *nœuds restants.*

(3) Pour chaque nœud n_i, *et chaque* $n_j \neq n_i$ *soit* $v_{(n_i s)n_j}$ *la valeur que* n_i *reçoit du* n_j *dans (2).*
(a) Si $\Big((v_{n_i s} \neq v_{(n_i s)n_j})$ *et (*n_i *n'a jamais accusé* s *et* n_j *ensemble) et* $(s \notin FAUTIFS_{n_i})$ *et* $(n_j \notin FAUTIFS_{n_i}))$ *alors*
accuser n_j *et* s.
(b) n_i *utilise la majorité de* $(v_{(n_1 s)n_i}, ..., v_{(n_{N-1} s)n_i})$.

(4) Pour chaque n_i,
répéter
recommencer= faux
Pour chaque nœud $n_j \neq n_i$
si $(n_j \notin FAUTIFS_{n_i})$ *et (règle de zèle fort ou faible s'applique)*
acquitter tous les nœuds qui ont été accusé à cause de n_j
ajouter n_j *à* $FAUTIFS_{n_i}$
recommencer = juste.
jusqu'à ce que (recommencer = faux).

TAB. 2.1 – *L'algorithme FIBA.*

2.4 Identification des nœuds fautifs en utilisant des messages signés

Nous avons vu qu'avec les messages signés, chaque nœud attache une signature digitale à son message. Dans l'algorithme des messages signés, proposé dans [13], la source envoie un message à chaque nœud. Chaque nœud ajoute sa signature à ce message et l'envoie aux autres nœuds. À leurs tours, ces derniers ajoutent leurs signatures et l'envoient aux autres nœuds, et ainsi de suite. Aucun nœud ne peut altérer le contenu du message d'un nœud non fautif sans que l'altération ne soit détectée par les autres nœuds non fautifs. Ainsi lorsqu'un nœud non fautif envoie un message aux autres nœuds, un nœud fautif ne peut pas altérer son message et l'envoyer aux autres nœuds. S'il tente de le faire, l'altération sera détectée immédiatement par le nœud receveur.

Il est évident qu'un algorithme d'identification des nœuds fautifs qui utilise les messages signés est simple. Cet algorithme ne requiert pas l'introduction de la notion de facteur d'accusation. En effet, considérons un système, dénoté $\mathcal{S}$, composé de N nœuds ($N > 3$), et notons par $\mathcal{OK}_{\mathcal{S}}$ l'ensemble des nœuds non fautifs de $\mathcal{S}$. De manière similaire, l'ensemble $\mathcal{FAUTIFS}_{\mathcal{S}}$ dénote les nœuds fautifs de $\mathcal{S}$. Ainsi, $\mathcal{FAUTIFS}_{\mathcal{S}} \cap \mathcal{OK}_{\mathcal{S}} = \phi$ et $\mathcal{FAUTIFS}_{\mathcal{S}} \cup \mathcal{OK}_{\mathcal{S}} = \mathcal{S}$. Voyons d'abord, la notion d'un nœud fautif:

Définition 1 *Un nœud n est dit fautif si l'une des actions suivantes est vraie:*

1. *le nœud n s'abstient ou omet d'envoyer des messages à au moins un autre nœud non fautif.*

2. *le nœud n altère le contenu d'un message d'au moins un autre nœud.*

Autrement, il est dit non fautif. □□□

Ainsi, un nœud n_i considère un autre nœud n_j comme fautif et l'ajoute à l'ensemble des nœuds fautifs identifiés par lui, dénoté $FAUTIFS_{n_i}$, si l'une des actions suivantes est vraie :

1. le nœud n_j s'abstient ou omet d'envoyer des messages à n_i.

2. le nœud n_j altère le contenu d'un message d'un autre nœud avant de l'envoyer à n_i.

Dans les deux sous-sections suivantes, nous allons prouver la validité de ce mécanisme d'identification des nœuds fautifs; ensuite, nous allons montrer que tous les nœuds fautifs seront identifiés.

2.4.1 Validité

Une condition requise pour la validité de n'importe quel mécanisme d'identification des nœuds fautifs : un nœud non fautif ne considère jamais un autre nœud non fautif comme fautif. Pour cela, il sera nécessaire de prouver le lemme suivant :

Lemme 1 $\forall n_i \in \mathcal{OK}_{\mathcal{S}},\ n_j \in \mathcal{OK}_{\mathcal{S}} \Longrightarrow n_j \notin FAUTIFS_{n_i}$.
Preuve :
Raisonnons par l'absurde en supposant que $n_j \in FAUTIFS_{n_i}$. *Par définition,* n_i *qui n'est pas fautif ne considère pas* n_j *comme fautif que si l'une des actions suivantes est vraie :*

1. n_j *a s'est abstenu ou a omis d'envoyer des messages à* n_i;

2. n_j *a altéré le contenu d'un message d'un autre nœud avant de l'envoyer à* n_i.

Supposons que n_j a commis l'une de ces actions, il s'en suit alors que $n_j \in \mathcal{FAUTIFS}_\mathcal{S}$. Mais ceci contredit l'hypothèse initiale : $n_j \in \mathcal{OK}_\mathcal{S}$. □□□

2.4.2 Identifaication de tous les nœuds fautifs

Dans cette sous-section, nous allons montrer que tous les nœuds fautifs vont être identifiés. Pour ce fait, nous allons prouver qu'un nœud fautif est considéré comme fautif par au moins un autre nœud non fautif.

Lemme 2 $\forall\, y \in \mathcal{S}, y \in \mathcal{FAUTIFS}_\mathcal{S} \implies \exists\, n_i$ *tel que* $y \in FAUTIFS_{n_i}$.
Preuve :
$y \in \mathcal{FAUTIFS}_\mathcal{S}$ *alors y est fautif. Par définition d'un nœud fautif, y a commis au moins l'une des acions suivantes :*

1. *le nœud y s'est abstenu ou a omis d'envoyer des messages à au moins un autre nœud.*
2. *le nœud y a altéré le contenu d'un message d'au moins un autre nœud.*

Ceci entraîne alors qu'il existe un nœud, disons n_i, tel que y s'est abstenu d'envoyer des messages à n_i, a omis d'envoyer des messages à n_i ou a altéré le contenu d'un message avant de l'envoyer à n_i. Mais alors, n_i aura détecté que y est fautif et $y \in FAUTIFS_{n_i}$.
□□□

Dans le lemme suivant, nous allons montrer que si un nœud non fautif n_i considère un autre nœud, soit y, comme fautif, alors le nœud y est en fait fautif.

Lemme 3 $\forall\, n_i \in \mathcal{OK}_\mathcal{S}, y \in FAUTIFS_{n_i} \implies y \in \mathcal{FAUTIFS}_\mathcal{S}$.
Preuve :

$y \in FAUTIFS_{n_i}$ alors y a commis l'une des actions suivantes :

1. *le nœud y s'est abstenu ou a omis d'envoyer des messages à n_i.*

2. *le nœud y a altéré le contenu d'un message avant de l'envoyer à n_i.*

D'après la définition 1, y est alors fautif, $y \in \mathcal{FAUTIFS}_S$. □□□

À partir des deux lemmes précédents, on peut obtenir le théorème suivant.

Théorème 1 $x \in \mathcal{FAUTIFS}_S \iff \exists\ n_i$ *tel que* $y \in FAUTIFS_{n_i}$ □□□

2.5 Illustration

Considérons le cas où $f_n = 2$ et $n = 7$. Nous allons décrire un exemple qui illustre l'application de nos règles pour identifier les nœuds fautifs durant les échanges des messages. Un nœud non fautif identifie les nœuds fautifs en examinant tous les messages reçus. En décrivant le mécanisme d'identification des nœuds fautifs du point de vue d'un nœud non fautif, il faut seulement examiner les messages reçus par ce nœud. La source est appelée n_0; les autres nœuds sont appelés respectivement : $n_1, n_2, n_3, n_4, n_5, n_6$. Initialement, la limite maximale des fautifs f_n est égale à deux. Les limites maximales des fautifs associées à chaque nœud sont aussi égale à deux ($f_{n_i} = 2$ pour $i = 1...6$). Les facteurs d'accusation, de tous les nœuds associés à tous les autres nœuds, sont initialement égaux à zéro ($fct_{n_i}^{n_j} = 0$ pour $i, j = 1...6$ et $j \neq i$). Supposons que les nœuds n_0 et n_1 sont fautifs.

Nous allons nous placer du point de vue du nœud n_3 qui n'est pas un fautif. Supposons que n_3 reçoive de la source n_0 la valeur $v_{3_0} = 0$. Dans le second tour d'échange des messages, chaque nœud envoie la valeur qu'il a reçue de la source à n_3 (figure 2.5).

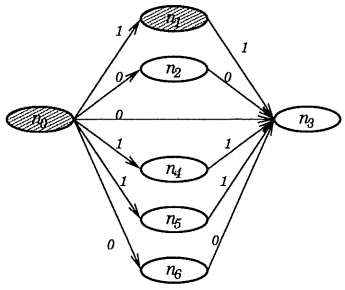


Figure 2.5 – 1^{er} *tour : Les valeurs de* n_0 *reçues par* n_3*, directement de* n_0 *et via tous les autres nœuds.*

Les valeurs reçues par n_3 via les autres nœuds sont : $v_{(3_0)_1} = 1$ via n_1, $v_{(3_0)_2} = 0$ via n_2, $v_{(3_0)_4} = 1$ via n_4, $v_{(3_0)_5} = 1$ via n_5 et $v_{(3_0)_6} = 0$ via n_6. La valeur v_{3_0} de la source reçue directement par n_3 est différente des valeurs de la source reçues par n_3 via n_1, n_4 et n_5. Le nœud n_3 accuse (n_0, n_1), (n_0, n_4) et (n_0, n_5). D'où les facteurs d'accusation de n_3 associés à chacun de ces nœuds seront : $fct_{n_3}^{n_0} = 3$, $fct_{n_3}^{n_1} = 1$, $fct_{n_3}^{n_4} = 1$ et $fct_{n_3}^{n_5} = 1$.

La règle de zèle fort s'applique, puisque $fct_{n_3}^{n_0} = 3 > f_{n_3} = 2$ et n_3 ajoute n_0 à $FAUTIFS_{n_3}$. Ainsi, le nœud n_3 décremente f_{n_3} de un ($f_{n_3} = f_{n_3} - 1 = 1$). Les nœuds n_1, n_4 et n_5 sont acquittés. leurs facteurs d'accusation seront décrementés de 1 ($fct_{n_3}^{n_1} = 0$, $fct_{n_3}^{n_4} = 0$ et $fct_{n_3}^{n_5} = 0$).

Dans le troisième tour d'échange des messages, le nœud n_3 reçoit des autres nœuds des rapports sur les valeurs envoyées par chacun des autres nœuds.

- Les rapports reçus par n_3 (figure 2.6) sur la valeur envoyée par n_1 :

  La valeur de n_1 reçue directement par n_3 est $v_{3_1} = 1$. Les rapports reçus par n_3 sur la valeur de n_1 via les autres nœuds sont : $v_{(3_1)_2} = 0$ via n_2, $v_{(3_1)_4} = 0$ via n_4, $v_{(3_1)_5} = 0$ via n_5 et $v_{(3_1)_6} = 0$ via n_6. La valeur du nœud n_1 v_{3_1} reçue directement

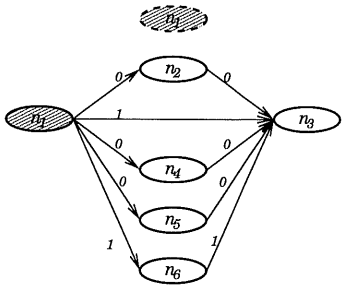


Figure 2.6 – 2^e*tour: Les valeurs de* n_1 *reçues par* n_3*, directement de* n_1 *et via tous les autres nœuds.*

par n_3 est différente des valeurs du nœud n_1 reçues par n_3 via n_2, n_4 et n_5. Le nœud n_3 accuse (n_1, n_2), (n_1, n_4) et (n_1, n_5). D'où les facteurs d'accusation de n_3 associés à chacun de ces nœuds seront: $fct_{n_3}^{n_1} = 3$, $fct_{n_3}^{n_2} = 1$, $fct_{n_3}^{n_4} = 1$ et $fct_{n_3}^{n_5} = 1$. La règle de zèle fort s'applique, puisque $fct_{n_3}^{n_1} = 3 > f_{n_3} = 1$ et n_3 ajoute n_1 à $FAUTIFS_{n_3}$ et $f_{n_3} = f_{n_3} - 1 = 0$. Les nœuds n_2, n_4 et n_5 sont acquittés. leurs facteurs d'accusation seront décrementés de 1.

- Les rapports reçus par n_3 sur la valeur envoyée par n_2 (figure 2.7) : le nœud n_3 reçoit une valeur directement de n_2 et via les autres nœuds. Cette valeur est la même pour tous les nœuds.

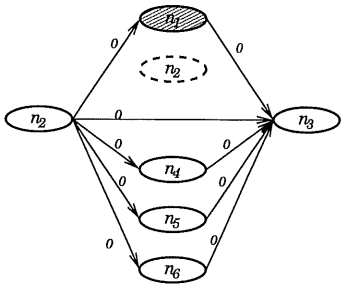


Figure 2.7 – 2^e*tour: Les valeurs de* n_2 *reçues par* n_3*, directement de* n_2 *et via tous les autres nœuds.*

– La figure 2.8 et la figure 2.9 nous montre les deux cas dans lesquels n_3 reçoit des valeurs et des rapports sur les valeurs de n_4 et n_5. C'est exactement le même scénario que le précédent.

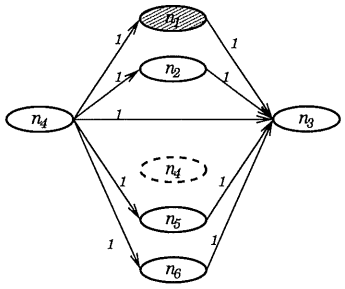


Figure 2.8 – 2^e*tour : Les valeurs de* n_4 *reçues par* n_3*, directement de* n_4 *et via tous les autres nœuds.*

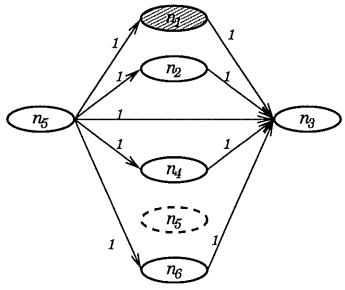


Figure 2.9 – 2^e*tour : Les valeurs de* n_5 *reçues par* n_3*, directement de* n_5 *et via tous les autres nœuds.*

– De même (figure 2.10) la valeur de n_6 reçue directement par n_3 est $v_{3_6} = 1$. La valeur de n_6 reçue par n_3 via les autres nœuds sont : $v_{(3_6)_1} = 0$, $v_{(3_6)_2} = 1$, $v_{(3_6)_4} = 1$ et $v_{(3_6)_5} = 1$. La valeur $v_{(3_6)_1} = 0$ est différente de v_{3_6} mais n_1 est considérée comme fautif par n_3. Ainsi aucune accusation ne se fait.

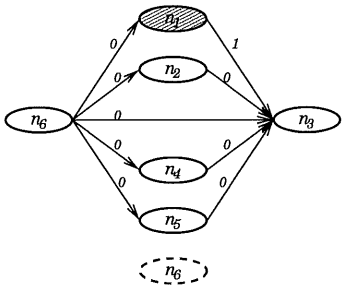


Figure 2.10 – 2^e*tour : Les valeurs de* n_6 *reçues par* n_3*, directement de* n_6 *et via tous les autres nœuds.*

2.6 Conclusion

Il est évident qu'avec la faute méchante il existe des cas où il est impossible d'identifier les nœuds fautifs qui travaillent de complicité avec une connaissance complète de l'état du système. Aucune hypothèse n'est prise sur leurs comportements et ils peuvent envoyer différentes valeurs aux autres nœuds. Dans certains modèles de fautes (arrêt de fonctionnement avec alerte, arrêt de fonctionnement sans alerte, faute d'omission et faute de synchronisation) il est possible de concevoir des mécanismes d'identification de tous les nœuds fautifs. De même, si on utilise certains modèles de réseaux, (*e.g.* le réseau de diffusion) qui ne permettent pas à un nœud fautif d'envoyer différentes valeurs pour les autres nœuds dans un même tour d'échange.

Notre mécanisme d'identification des nœuds fautifs n'a rien ajouté au coût de la résolution de BA, soit de par ses communications, soit par l'exigence du système. Nous avons pris l'algorithme OM tel qu'il est, et nous avons fourni un mécanisme d'identification des nœuds fautifs en examinant les messages reçus par chaque nœud non fautif.

CHAPITRE 3

Identification des nœuds et des lignes de communication fautifs pour le problème Byzantin dans un réseau de diffusion généralisé

3.1 Introduction

Un système distribué est composé d'un ensemble des nœuds autonomes qui communiquent entre eux via un réseau. Pour coopérer, ces nœuds doivent coordonner leurs activités. La diffusion est le paradigme de communication qui permet à un nœud de communiquer une valeur locale à tous les autres nœuds dans le système. Le problème de diffusion posé peut être résumé comme suit : soit un nœud distingué, appelé source, qui

diffuse une valeur locale. On doit alors satisfaire les conditions suivantes :

1. tous les nœuds non fautifs adoptent une même valeur;
2. si la source est non fautive, alors tous les nœuds non fautifs adoptent sa valeur.

Ces deux conditions sont identiques à celles formulées par le *BA*, [13]. Notre objectif est de développer un algorithme optimal pour atteindre un accord et identifier les composants fautifs tout en tolérant un nombre maximum de ces composants.

Dans un système distribué utilisant un réseau connecté complet, les fautes méchantes des nœuds sont pénibles pour plusieurs raisons [2]:

1. un nœud fautif peut envoyer des informations incohérentes à différents nœuds.
2. à moins que le graphe de communication ne soit complètement connecté, les nœuds fautifs peuvent altérer les messages circulant sur le réseau.
3. les nœuds fautifs peuvent générer de faux messages en prétendant les envoyer au nom d'autres nœuds.

Nous avons vu dans le chapitre précédent qu'il est impossible de développer un algorithme qui résout le *BA* et identifie les composants fautifs dans le système sans restreindre l'habileté de ces composants. Le comportement visible d'un nœud fautif peut être effectivement restreint par l'utilisation de certains mécanismes. Par exemple, si le système supporte un mécanisme d'authentification comme les messages signés, les messages altérés par les nœuds fautifs seront détectés. Ainsi, de faux messages ne pourront être spontanément générés. De cette restriction sur le comportement d'un nœud fautif, il résulte des algorithmes qui sont plus simples et qui peuvent tolérer plus de fautes que leur contrepartie sans authentification. Mais l'interaction des composants fautifs reste un obstacle devant

le développement d'un algorithme optimal d'identification des composants fautifs.

Dans ce qui suit on propose une architecture de communication [21], autre que le réseau connecté complet point à point. Cette architecture, fondée sur les réseaux de diffusion, contribue à restreindre le comportement des nœuds fautifs. En se basant sur cette architecture, nous allons développer un algorithme qui résout le *BA* et qui identifie les composants fautifs dans le système.

Ce chapitre est organisé de la façon suivante: la section suivante est consacrée à la conception d'un modèle de réseau de diffusion efficace et pratique. La section 3.3 est une description des propriétés globales du système et des hypothèses élaborés. Les modèles des fautes des nœuds et des lignes des communication sont discutés. Dans la section 3.4, nous proposons un algorithme, appelé *FIGBA* (*Fault Identification for Generalized Byzantine Agreement*). Cet algorithme résout le *BA* et identifie tous les composants fautifs dans le système (nœuds et lignes de communication).

3.2 Modèles des réseaux de diffusion

Cette section est consacrée à la conception d'un modèle d'un réseau de diffusion efficace et pratique. Les principaux composants de communication dans le réseau de diffusion seront identifiés. Nous présenterons ensuite le réseau de diffusion et le réseau de diffusion redondant. Ces deux modèles serviront à la conception du réseau de diffusion généralisé. Ce dernier est fondé sur les réseaux de diffusions et combine le réseau connecté complet et le réseau de diffusion. Ce modèle sera utilisé comme architecture de communication pour le *FIGBA*.

3.2.1 Principaux composants de communication dans un réseau de diffusion

Les réseaux de diffusion sont devenus une topologie populaire pour les réseaux locaux. Les raisons de cette popularité sont liés à la simplicité, la fiabilité et la performance élevée. L'exemple le plus connu pour ce modèle de réseaux est peut-être l'éthernet. La figure 3.1 montre une configuration typique d'un système distribué basée sur un réseau de diffusion.

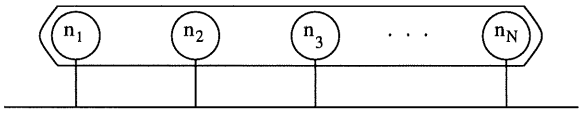


Figure 3.1 – *Réseau de diffusion.*

Le composant central d'un réseau de diffusion est le canal. Typiquement, c'est un moyen de transmission passif comme un câble coaxial, un câble en fibre optique, ou simplement un espace (dans le cas d'un réseau de diffusion d'une radio). Les nœuds sont connectés au canal par des liens de communication. Ces liens contiennent des composants actifs comme le signal d'émetteur-récepteur, les unités d'interface de réseau et la mémoire tampon. L'accès au canal par les liens de communication est contrôlé par un protocole de transmission à accès multiple. Nous notons que les protocoles de transmission sans collision existent pour les canaux à accès multiple.

Chaque réseau de diffusion est connecté à une passerelle (*Gateway*). Cette dernière est nécessaire pour le fonctionnement d'un réseau de diffusion [19]. Elle sert aussi d'intermediaire de sécurité.

3.2.2 Réseau de diffusion et réseau de diffusion redondant

L'opération fondamentale de communication pour un réseau de diffusion est **broadcast**(v), où v est le contenu du message communiqué. Par l'hypothèse d'authentification, un nœud recevant un message est capable d'identifier d'une façon fiable le diffuseur sous n'importe quel modèle de fautes de nœuds.

Avec des lignes de communication non fiables, l'opération **broadcast**(v) ne peut garantir que tous les nœuds ont reçus la valeur v diffusée. Pour accroître la fiabilité du réseau, on a proposé dans [1], un réseau de diffusion *R-redondant* généré en reproduisant chaque composant de communication R fois. Comme l'indique la figure 3.2, la construction exige R réseaux de diffusion indépendants où chacun des nœuds est connecté à chacun de R canaux par un lien indépendant.

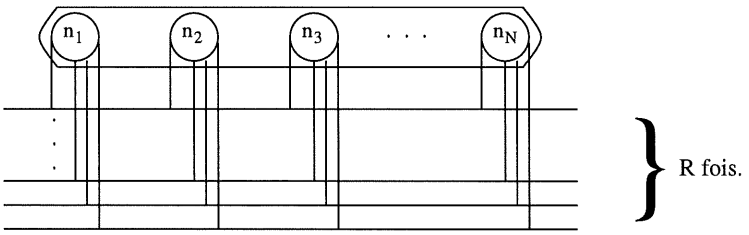


Figure 3.2 – *Réseau de diffusion R-redondant.*

Compte tenu du fait que chaque nœud, dans un réseau de diffusion *R-redondant*, est connecté à R canaux, on augmente la syntaxe primitive de diffusion de la façon suivante : **broadcast**(v, i) où v est le message à communiquer, et $i \in \{1, 2, ..., R\}$ dénote le canal.

Nous allons utiliser la syntaxe suivante :

$$\begin{aligned}
\textbf{broadcast}(v, *) \equiv & \\
& \text{for each } i \text{ in } \{1, 2, ..., R\} \\
& \textit{do} \\
& \quad \textbf{broadcast}(v, i) \\
& \textit{end}
\end{aligned}$$

Notons que cette opération ne doit pas être absolument atomique. De plus, on ne pose aucune hypothèse sur l'ordre ou le réglage relatif de la diffusion sur les divers canaux. Aussi, aucune hypothèse n'est élaborée sur le comportement d'un nœud fautif qui exécute l'opération **broadcast**$(v, *)$. Ainsi, ce nœud peut diffuser uniquement sur certains canaux ou encore diffuser des valeurs incohérentes.

Le réseau de diffusion est facile à implanter. Nous avons vu dans la section 3.2.1 qu'une passerelle est nécessaire pour le fonctionnement de ce réseau. Malheureusement, si ce composant tombe en panne, le réseau cesse de fonctionner. De là, on peut dire qu'un réseau de diffusion n'est pas pratique. Dans la section suivante nous allons présenter un réseau, proposé dans [21], dont le principe est proche du principe de réseau de diffusion et qui ne requiert pas une passerelle.

3.2.3 Réseau de diffusion généralisé

Un réseau de diffusion généralisé combine le principe d'un réseau de diffusion et celui d'un réseau connecté complet. Les n nœuds sont répartis en γ groupes et chaque groupe contient μ nœuds. Il existe un lien entre toute paire de groupes. Les avantages tirés d'un

réseau donné dépendent de plusieurs facteurs dont :

1. Si le nombre des nœuds dans le système croît, on aura moins de ports d'entrée/sortie et le trafic de communication décroît.

2. À cause des limitations du trafic de communication sur une ligne de communication, le nombre des nœuds connectés à un canal ne peut pas être plus grand qu'une constante, soit 2μ [18, 22].

3. Le nombre de ports d'entrée/sortie qui sont alloués à un nœud est limité [18, 22]. En effet, ce nombre ne peut pas être plus grand qu'une constante, soit $\gamma - 1$.

Soit un système composé de n nœuds, examinons l'applicabilité des réseaux de diffusion et les réseaux connectés complets sous l'influence de ces trois propriétés. Nous avons :

- les réseaux de diffusion ne satisfont pas (1) et (2) si $n > 2\mu$.

- les réseaux connectés complets sont inapplicables à cause de (1) et (3) si $n > \gamma$.

Dans [21], on a organisé les nœuds en plusieurs groupes afin que les trois propriétés précédentes (1), (2) et (3) soient satisfaites :

- les nœuds sont regroupés exactement en γ groupes;

- chaque groupe contient exactement μ nœuds.

À titre d'exemple, les figures 3.3 et 3.4 montrent un réseau de diffusion généralisé formé par quatre groupes $G1$, $G2$, $G3$ et $G4$ dont chaque groupe contient cinq nœuds. Les nœuds dans chaque groupe G_i $(1 \leq i \leq 4)$ sont nommés n_{ij} pour $1 \leq j \leq 5$. Les groupes G_1, G_2, G_3 et G_4 sont complètement connectés les uns aux autres par des canaux comme

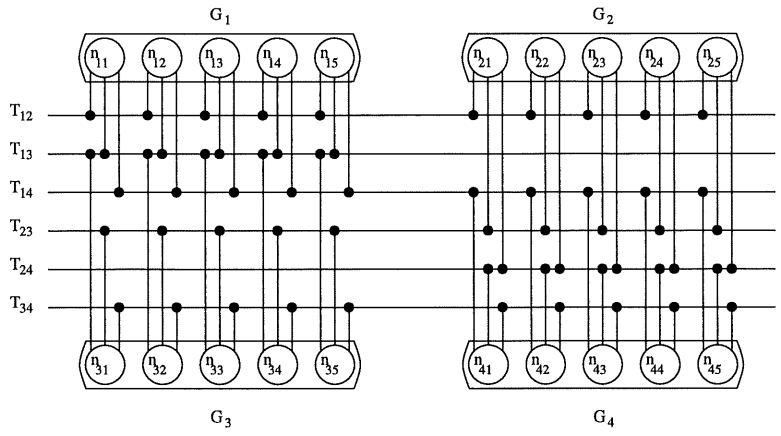


Figure 3.3 – *La configuration d'un réseau de diffusion généralisé formé de quatre groupes dont chaque groupe contient cinq nœuds*

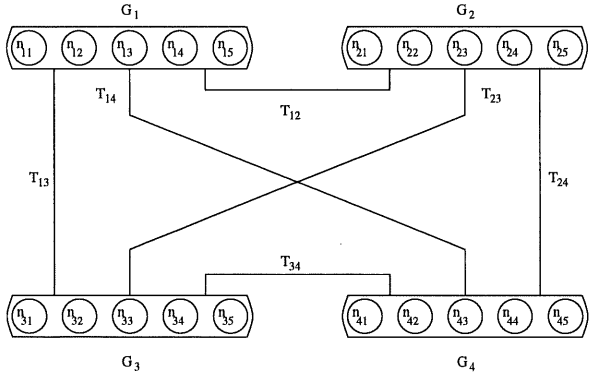


Figure 3.4 – *La connexion entre les groupes d'un réseau de diffusion généralisé formé de quatre groupes dont chaque groupe contient cinq nœuds.*

le montre la figure 3.4. Dans ce graphe complet, chaque nœud requiert seulement trois ports d'entrée/sortie et six canaux et aucune passerelle n'est requis.

Il n'existe qu'un lien unique entre toute paire de groupes. Ainsi, le comportement d'un nœud fautif devient alors partiellement limité. En effet, l'unicité des liens entre toute paire de groupes élimine les messages incohérents entre les nœuds de ces deux groupes. Mais un nœud fautif peut diffuser uniquement sur certains canaux ou encore diffuser des valeurs incohérentes pour différents groupes. D'autre part, les messages ne peuvent pas être altérés car les nœuds ne sont pas sur la route des messages, chaque nœud est connecté à un canal par son propre lien. Le message envoyé est transféré entre le nœud source et le canal avant d'être délivré aux autres nœuds. Le canal est un bus commun entre un ensemble des nœuds et les nœuds ne peuvent pas altéré un message dans le canal.

Chaque groupe est connecté à un autre groupe par un canal. Un nœud diffuseur dans un groupe doit exécuter l'opération **broadcast**(v, i). Par exemple, l'opération **broadcast**$(v, 2)$ exécutée par n_{11} veut dire que n_{11} diffuse la valeur v au groupe G_2.

Grâce à cette architecture, les messages envoyés aux autres groupes par un nœud dans un groupe donné, soit K, sont reçus par les autres nœuds du groupe K. Physiquement, ceci est dû au bus commun entre les nœuds d'un groupe et les nœuds de chaque autre groupe. Par exemple, les nœuds n_{11}, n_{12}, n_{13}, n_{14} et n_{15} du groupe G_1 sont connectés à tous les autres groupes par les canaux : T_{12}, T_{13} et T_{14} (figure 3.4). Si n_{11} envoie un message au groupe G_2, en plus des nœuds du G_2, les autres nœuds du groupe G_1 (n_{12}, n_{13}, n_{14} et n_{15}) reçoivent également ce message. De même, si n_{11} envoie un message aux groupes G_3 et G_4, les autres nœuds du groupe $G1$ reçoivent également ce message.

Si le nombre de nœuds est cinquante, chaque nœud requiert seulement quatre ports d'en-

trée/sortie et dix canaux dans un réseau de diffusion généralisé formé de cinq groupes dont chaque groupe contient dix nœuds. La table 3.1 montre le nombre de ports d'entrée/sortie, de lignes de communication et de passerelles requis pour différents réseaux dans un système composé de cinquante nœuds.

Type de réseau Propriétés	connecté complet	diffusion	diffusion généralisé (cinquante nœuds répartis dans cinq groupes dont chaque groupe contient dix nœuds)
Nombre de ports d'entrée/sortie par nœud	49	1	4
Nombre total de lignes de communication	1225	2	10
Nombre de passerelle	Nul	1	Nul

TAB. 3.1 – *Ports d'entrée/sortie, lignes de communication et passerelles requis par les différents réseaux dans un système composé de cinquante nœuds.*

3.3 Modèle de fautes

Les composants principaux sont les nœuds, les canaux et les liens qui connectent les nœuds aux canaux. Les composants qui peuvent être fautifs sont les nœuds et les canaux. Dans le cas où un lien est non fautif, il transfert simplement les messages entre le canal et son nœud associé. Une faute dans les liens cause quelques fois la perte des messages arrivant au nœud associé ou partant de ce dernier. On ne peut pas distinguer entre un lien fautif ou un nœud fautif avec lequel il est connecté. En effet, considérons le cas d'un nœud n_{iK} qui ne reçoit pas une valeur attendue d'un autre nœud n_{jL} via un canal non fautif. Le nœud n_{iK} ne peut pas savoir si c'est le lien qui a entraîné la perte du message ou bien c'est le nœud n_{jL} qui n'a pas envoyé un message. Pour cela, si le lien est fautif, le nœud connecté à ce lien sera considéré comme fautif. Les nœuds et les canaux

peuvent être fautifs.

Le modèle de faute considéré est le suivant:

- la faute méchante pour les nœuds.
- la faute d'arrêt de fonctionnement sans alerte pour les canaux.

Évidemment un système qui peut tolérer les fautes méchantes peut aussi tolérer les fautes d'omission ou les fautes d'arrêt de fonctionnement sans alerte. Il est à noter que les autres types de fautes pour les canaux sont extrêmement rares [22]. En effet, les méthodes de correction d'erreurs, de détection d'erreurs ou de détection et de correction d'erreurs sont inclus dans le protocole [18].

3.4 Algorithme *FIGBA* (*Fault Identification for Generalized Byzantine Agreement*)

La plupart des formulations précédentes du *BA* utilisent comme architecture de communication un réseau connecté complet fiable. Dans cette section, nous présentons un algorithme efficace qui implante le *BA* dans un réseau de diffusion généralisé en présence des fautes méchantes pour les nœuds et des fautes d'arrêt de fonctionnement sans alerte pour les canaux. Le but de *FIGBA* est de:

1. résoudre le *BA*.
2. identifier les composants fautifs dans le système, peu importe si ces composants sont parmi les nœuds ou les canaux.

Dans la suite, nous allons tout d'abord donner les définitions et les notations préliminaires. Les hypothèses élaborées par le *FIGBA* sont discutées dans la section 3.4.2. Les sections 3.4.3 et 3.4.4 sont consacrées à analyser le nombre des composants fautifs alloués par le *FIGBA* et à décrire les tours requis. Dans la section 3.4.5, nous décrivons le principe de l'identification des composants fautifs pour le *BA* généralisé *FIGBA*.

3.4.1 Définitions et notations

Voici quelques définitions et notations utilisées dans la suite :

- γ : nombre des groupes de nœuds dans le réseau.
- μ : nombre des nœuds dans chaque groupe.
- n : nombre total des nœuds dans le système, on a ainsi $n = \gamma\mu$.
- $\mathcal{S}$: l'ensemble de tous les nœuds.
- s : la source donnée.
- n_{iK} : le $i^{\text{ème}}$ nœud du groupe K; n_{iS} : le $i^{\text{ème}}$ nœud du groupe source.
- T_{LK} : le canal qui connecte le groupe L et le groupe K; T_{SK} : le canal qui connecte le groupe source et le groupe K.
- $V_{n_{iK}}(n_{jL})$ est l'ensemble des valeurs envoyées par le nœud n_{jL} et reçues par le nœud n_{iK}.
- $V_{n_{iK}}(L)$ est l'ensemble des valeurs envoyées par tous les nœuds dans le groupe L et reçues par le nœud n_{iK}.

- $FAUTIFS_{n_{iK}}$: l'ensemble des composants considérés comme fautifs par le nœud n_{iK}.
- $ABSTAIN_{n_{iK}}(T_{K.})$: l'ensemble des nœuds qui se sont abstenus d'envoyer des messages au nœud n_{iK} via le canal $T_{K.}$.

Définition 2 *Soit n_{jL} et n_{iK} deux nœuds, $V_{n_{iK}}(n_{jL})$ est l'ensemble des valeurs envoyées par le nœud n_{jL} et reçues par le nœud n_{iK}. On dit que $V_{n_{iK}}(n_{jL})$ est consistant en la valeur du nœud n_{jL}, dénoté $v_{n_{jL}}$, si et seulement si*

- *$V_{n_{iK}}(n_{jL})$ consiste en une seule valeur $v_{n_{jL}}$, la valeur envoyée par le nœud n_{jL}; ou*
- *$V_{n_{iK}}(n_{jL})$ consiste en deux valeurs : la valeur nulle ϕ et $v_{n_{jL}}$, la valeur envoyée par le nœud n_{jL}.*

□□□

Définition 3 *La valeur de défaut est une valeur prédéfinie par le protocole.* □□□

Définition 4 *La valeur majoritaire d'un ensemble est la valeur qui se répète le plus souvent dans cet ensemble. Si cette valeur n'existe pas, la valeur de défaut sera pris.* □□□

3.4.2 Hypothèses

Le *FIGBA* exige les hypothèses[1] suivantes :

1. Un système synchrone qui contient n nœuds dont au plus f_n nœuds peuvent être fautifs, sans causer une panne dans le fonctionnement du système. Le réseau garantit un temps limité de livraison de messages pour tous les nœuds.

1. Notons que les trois premières sont similaires aux hypothèses élaborées sur le *BA* [13].

2. Chaque nœud possède une horloge et les nœuds non fautifs maintiennent un temps local qui ne diffère pas de plus d'une constante connue.

3. L'envoyeur d'un message est toujours identifiable par le récepteur.

4. Une faute dans le canal entraîne la perte de tous les messages arrivants. Par définition d'un réseau de diffusion généralisé, on a deux cas :

   (a) tous les nœuds connectés à un canal (dans le cas où ce canal est non fautif) reçoivent le message diffusé.

   (b) aucun des nœuds connectés à un canal (dans le cas où ce canal est fautif) n'a reçu le message diffusé.

5. Seuls les nœuds et les canaux peuvent être fautifs.

6. Un canal fautif ne peut spontanément générer des messages.

7. Le nombre de nœuds fautifs dans chaque groupe doit être plus grand que le nombre des nœuds non fautifs.

Remarquons que toutes ces hypothèses peuvent être respectées dans le cadre des réseaux de diffusion généralisé. Le comportement d'un nœud fautif est restreint par l'utilisation de ce modèle des réseaux. Physiquement c'est impossible pour un nœud d'envoyer des messages incohérents pour différents nœuds dans la même émission (**broadcast**). Ce comportement est assuré même si le modèle de fautes de nœuds est la faute méchante.

3.4.3 Tours requis par *FIGBA*

Il est habituel de mesurer la complexité temporelle en terme de *tours*. Chaque tour consiste en une ou plusieurs étapes d'échange de messages et de calcul. Durant un tour,

chaque nœud peut communiquer avec tous les autres nœuds dans le système. Toutefois, les messages envoyés par un nœud dans un tour donné ne peuvent pas dépendre des messages reçus durant ce même tour.

L'algorithme proposé *FIGBA* requiert cinq tours. Les deux premiers tours consistent en des échanges de messages pour que tous les nœuds puissent adopter la même valeur. On exige le troisième tour pour faire face aux effets des canaux fautifs. En effet, ce tour permet à tous les groupes qui sont connectés au groupe source par un canal fautif[2] de recevoir de valeurs via les canaux non fautifs à travers les autres groupes. Le troisième tour est requis pour que les nœuds du groupe K puissent recevoir des valeurs via des canaux non fautifs à travers les autres groupes. Lors des deux derniers tours, les nœuds s'échangent leurs listes des composants fautifs. Ces échanges permettent aux nœuds de se mettre d'accord sur une liste qui contient tous les composants fautifs dans le système. Le cinquième tour est requis pour faire face aux effets des canaux fautifs. Chaque tour consiste en les actions suivantes :

`tour 1` : la source diffuse sa valeur à tous les autres nœuds.

`tour 2` : chaque nœud du groupe source diffuse à tous les autres nœuds la valeur de la source reçue au premier tour.

`tour 3` : chaque nœud, n'appartenant pas au groupe source, diffuse à tous les autres nœuds la valeur qu'il a adoptée au second tour.

`tour 4` : chaque nœud diffuse à tous les autres nœuds son ensemble des composants fautifs. Chaque nœud collecte tous les ensembles envoyés pour construire un nouveau ensemble des composants fautifs. Cet ensemble doit contenir les composants fautifs

2. Remarquons que dans les deux premiers tours, seulement les nœuds du groupe source diffusent des valeurs

de tous les groupes.

`tour 5` : chaque nœud diffuse à tous les autres nœuds l'ensemble des composants fautifs qu'il a adopté au quatrième tour.

3.4.4 Nombre des composants fautifs alloués par *FIGBA*

Les nœuds

Durant les tours d'échanges de messages et des listes des composants fautifs, toutes les décisions sont prises au niveau du groupe d'une façon majoritaire. Pour cela, le nombre des nœuds fautifs dans chaque groupe doit être strictement inférieur au nombre des nœuds non fautifs dans le même groupe. Ceci entraîne que la majorité des nœuds dans le système soient non fautifs. Rappelons que μ est le nombre des nœuds dans un groupe, Le nombre maximum des nœuds qui peuvent être fautifs dans un groupe est $\frac{(\mu-1)}{2}$ (figure 3.5). Notons par f_n le nombre maximum des nœuds qui peuvent être fautifs dans le système et rappelons que γ est le nombre des groupes, alors :

$$f_n \leq \frac{(\mu-1)}{2}\gamma$$

Les canaux

Le graphe des connexions entre les groupes est complet (figure 3.5). Chaque groupe est connecté aux autres groupes par $\gamma - 1$ canaux. Pour assurer qu'un groupe ait au moins un canal non fautif, nous exigeons que le nombre maximum des canaux qui peuvent être fautifs ne dépasse pas $\gamma - 2$. Notons par f_c ce nombre, nous avons ainsi :

$$f_c \leq \gamma - 2$$

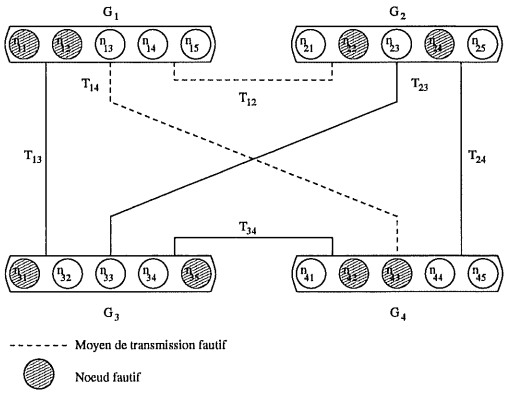


Figure 3.5 – *Nombre maximum de nœuds et de canaux fautifs dans un système composé de vingt nœuds répartis en quatre groupes ($\gamma = 4$ et $\mu = 5$).*

Dans la figure 3.5, on peut remarquer que chaque groupe peut contenir, au maximum, deux nœuds fautifs ($f_n = 8$). Le nombre maximum des canaux qui peuvent être fautifs est égal à deux ($f_c = 2$).

3.4.5 Principe d'identification des composants fautifs

Avant de discuter l'algorithme, nous allons présenter les règles d'identification des composants fautifs. Chaque nœud non fautif est chargé d'identifier tous les composants fautifs dans son groupe. Grâce à l'architecture d'un réseau de diffusion généralisé, chaque nœud n_{iK} du groupe K reçoit toutes les valeurs envoyées par n'importe quel autre nœud n_{jK} du groupe K à tous les autres groupes via les différents canaux connectés au groupe K.

Dans chaque tour, chaque nœud utilise les règles suivantes pour identifier les composants fautifs :

1. si n_{iK} ne reçoit pas de messages via un canal, alors n_{iK} ajoute ce canal à $FAUTIFS_{n_{iK}}$.

2. si n_{iK} reçoit au moins une valeur via un canal, il ajoute tous les nœuds qui n'ont pas envoyé des valeurs via ce canal à $FAUTIFS_{n_{iK}}$.

3. si n_{iK} reçoit des valeurs différentes de n_{jK}, alors n_{iK} ajoute n_{jK} à $FAUTIFS_{n_{iK}}$. En effet, dans une telle situation on dit que l'ensemble des valeurs $V_{n_{iK}}(n_{jK})$ n'est pas consistant en la valeur de n_{jK} (Définition 1).

Le *FIGBA* requiert cinq tours. Au premier tour, tous les nœuds du groupe source adoptent une même valeur. En effet, Grâce à l'architecture du réseau de diffusion généralisé, tous les messages envoyés par la source aux autres groupes sont reçus par les autres nœuds du groupe source. Évidemment, si la source diffuse des valeurs incohérentes, elle sera identifiée par les nœuds du groupe source. Au second tour, tous les nœuds de tous les groupes adoptent une même valeur, à l'exception de ceux qui sont connectés au groupe source par un canal fautif. Les nœuds fautifs du groupe source et les canaux fautifs connectés au groupe source seront identifiés par au moins les nœuds non fautifs du groupe source. Au troisième tour, tous les nœuds non fautifs de tous les groupes adoptent une même valeur. Si la source est non fautive, tous les nœuds non fautifs de tous les groupes adoptent sa valeur. Ainsi le *BA* est résolu. Tous les nœuds fautifs et tous les canaux fautifs seront identifiés par au moins les nœuds non fautifs de leurs groupes. Au quatrième tour, tous les nœuds non fautifs identifient tous les nœuds fautifs et tous les canaux fautifs, à l'exception ceux qui sont connectés par des canaux fautifs. Finalement au cinquième tour, tous les nœuds non fautifs identifient tous les nœuds fautifs et tous les canaux fautifs. Ainsi, le *BA* est résolu et tous les composants fautifs sont identifiés par les nœuds non fautifs.

3.4.6 Tour 1

Dans ce tour la source diffuse sa valeur à tous les autres nœuds y compris bien sûr les nœuds de son propre groupe. Après cette diffusion, chacun des nœuds doit décider de la valeur qu'il va adopter. Finalement, les nœuds du groupe source doivent décider si la source est fautive ou non.

Étape de diffusion/réception : la source, dénotée s, diffuse sa valeur via les différents canaux connectés au groupe source, dénotée S. Notons que les nœuds du groupe source S reçoivent une copie de toutes les valeurs que la source s envoie aux nœuds des autres groupes via les différents canaux. Au contraire, chaque nœud d'un autre groupe reçoit au plus une valeur de la source s via le canal qui connecte son groupe avec le groupe source S. Après cette diffusion, le traitement suivant est effectué :

1. pour chaque nœud n_{iS}, appartenant au groupe source S,
   - si n_{iS} a reçu une valeur de la source s, il ajoute alors cette valeur à $V_{n_{iS}}(s)$, l'ensemble des valeurs reçues par n_{iS} et envoyées par s;
   - si n_{iS} n'a pas reçu de valeur via un canal quelconque, disons $T_{S.}$, alors il ajoute ϕ, la valeur nulle, à $V_{n_{iS}}(s)$, l'ensemble des valeurs reçues par n_{iS} et envoyées par la source s. Il ajoute aussi la source s à $ABSTAIN_{n_{iS}}(T_{S.})$, l'ensemble des nœuds qui n'ont pas envoyé une valeur via le canal $T_{S.}$.
2. pour chaque nœud n_{iK}, appartenant à un groupe dénoté K autre que le groupe source S,
   - si n_{iK} a reçu une valeur de la source s, il ajoute cette valeur à $V_{n_{iK}}(s)$, l'ensemble des valeurs reçues par n_{iK} et envoyées par s;
   - si n_{iK}, n'a pas reçu de valeur via T_{SK}, alors il ajoute ϕ, la valeur nulle, à $V_{n_{iK}}(s)$, l'ensemble des valeurs reçues par n_{iK} et envoyées par la source

s. Il ajoute aussi la source s à $ABSTAIN_{n_{iK}}(T_{SK})$, l'ensemble des nœuds qui n'ont pas envoyé une valeur via le canal T_{SK}.

Étape de décision : chaque nœud dans le système est conduit à prendre une ou plusieurs décisions. Le traitement suivant est donc effectué :

1. pour chaque nœud n_{iS}, appartenant au groupe source S, il procède comme suit :
   - s'il existe une valeur majoritaire[3] pour les valeurs de $V_{n_{iS}}(s)$, alors n_{iS} adopte cette valeur. Sinon, n_{iS} adopte la valeur de défaut[4].
   - si $V_{n_{iS}}(s)$, l'ensemble des valeurs reçues par n_{iS} et envoyées par s, n'est pas consistant[5] par rapport à la valeur de s, alors n_{iS} ajoute s à $FAUTIFS_{n_{iS}}$.
2. pour chaque nœud n_{iK}, appartenant à un groupe K autre que le groupe source S, n_{iK} adopte la valeur appartenant à $V_{niK}(s)$.

3. la valeur qui se répète le plus souvent
4. la valeur prédéfinie par le protocole.
5. La consistance d'un ensemble dans une valeur donnée est définie dans la section 3.4.1. Intuitivement, $V_{n_{iS}}(s)$ est consistant par rapport à la valeur de s si et seulement si (1) $V_{n_{iS}}(s)$ consiste en la seule valeur de s; ou (2) $V_{n_{iS}}(s)$ consiste en deux valeurs : la valeur nulle ϕ et la valeur de s.

Round 1

begin **diffusion step**
    begin source node
        Broadcast(v_s, *);
    end source node
end **diffusion step**

begin **reception step**
    for each node $n_{iS} \in S$, $n_{iS} \neq s$
        if n_{iS} has received a value from s by T_{SK}, $(1 \leq K \leq \gamma \ \& \ K \neq S)$ *then*
            $V_{n_{iS}}(s) \leftarrow V_{n_{iS}}(s) \bigcup \{v_s\}$;
        else
            $V_{n_{iS}}(s) \leftarrow V_{n_{iS}}(s) \bigcup \{\phi\}$;
            $ABSTAIN_{n_{iS}}(T_{SK}) = \{s\}$;
        end
    end for
    for each node $n_{iK} \in K$, $K \neq S$
        if n_{iK} has received a value by T_{SK} *then*
            $v_{n_{iK}} \leftarrow v_s$;
        else
            $v_{n_{iK}} \leftarrow \phi$;
            $ABSTAIN_{n_{iK}}(T_{SK}) = \{s\}$;
        end
    end for
end **reception step**

begin **decision step**
    for each node $n_{iS} \in S$, $n_{iS} \neq s$
        if $V_{n_{iS}}(s)$ is not consistent w.r.t. v_s *then*
            $FAUTIFSn_{iS} \leftarrow s$;
        end
        $v_{n_{iS}} = maj(V_{n_{iS}}(s))$;
    end for
end **decision step**

TAB. 3.2 – *L'algorithme FIGBA: tour 1.*

`Remarque:` La fonction majorité, dénotée $maj(V_{n_{iS}}(s))$ et utilisée dans cet algorithme, retourne la valeur majoritaire si elle existe, sinon elle retourne la valeur de défaut.

3.4.7 Tour 2

Pendant ce tour, chacun des nœuds, appartenant au groupe source S, diffuse à tous les autres nœuds la valeur qu'il a adoptée au premier tour. Chaque nœud n'appartenant pas

au groupe source S doit décider quant à la valeur qu'il va adopter en se basant sur la valeur reçue au premier tour et sur les valeurs qu'il va recevoir pendant ce second tour. Tous les nœuds sont chargés d'identifier les canaux fautifs. Chaque nœud appartenant au groupe source doit aussi identifier les nœuds fautifs dans son groupe.

Étape de diffusion/réception : chaque nœud n_{iS} qui appartient au groupe source S, autre que la source s, se comporte comme une source et diffuse la valeur qu'il a adoptée au premier tour. Rappelons que lorsqu'un nœud du groupe source S diffuse une valeur aux autres nœuds via différents canaux, les nœuds de S reçoivent une copie de toutes les valeurs que le nœud n_{iS} envoie aux nœuds des autres groupes via les différents canaux. En conséquence, chaque nœud d'un autre groupe reçoit au plus une valeur du nœud n_{iS} via le canal qui connecte son groupe avec le groupe source S. Après cette diffusion, le traitement suivant est effectué :

1. pour chaque nœud n_{jS}, appartenant au groupe source S,
   - si n_{jS} a reçu une valeur de la source n_{iS}, il ajoute alors cette valeur à $V_{n_{jS}}(n_{iS})$, l'ensemble des valeurs reçues par n_{jS} et envoyées par n_{iS};
   - si n_{jS} n'a pas reçu de valeur via un canal quelconque, disons $T_{S.}$, alors il ajoute ϕ, la valeur nulle, à $V_{n_{jS}}(n_{iS})$, l'ensemble des valeurs reçues par n_{jS} et envoyées par la source n_{iS}. Il ajoute aussi la source n_{iS} à $ABSTAIN_{n_{jS}}(T_{S.})$, l'ensemble des nœuds qui n'ont pas envoyé une valeur via le canal $T_{S.}$.
2. pour chaque nœud n_{iK}, appartenant à un groupe dénoté K autre que le groupe source S,
   - si n_{iK} a reçu une valeur de la source n_{iS}, il ajoute cette valeur à $V_{n_{iK}}(S)$, l'ensemble des valeurs reçues par n_{iK} et envoyées par les nœuds du S;

- si n_{iK}, n'a pas reçu de valeur via T_{SK}, alors il ajoute ϕ, la valeur nulle, à $V_{n_{iK}}(S)$, l'ensemble des valeurs reçues par n_{iK} et envoyées par les nœuds du groupe source S. Il ajoute aussi la source n_{iS} à $ABSTAIN_{n_{iK}}(T_{SK})$, l'ensemble des nœuds qui n'ont pas envoyé une valeur via le canal T_{SK}.

Étape de décision : lorsque les étapes de diffusion ont été effectués par chaque nœud du groupe source S, le traitement suivant est effectué :

1. pour chaque nœud n_{jS}, appartenant au groupe source S, il procède comme suit :
   - si $V_{n_{jS}}(n_{iS})$, l'ensemble des valeurs reçues par n_{jS} et envoyées par le nœud n_{iS}, n'est pas consistant par rapport à la valeur de n_{iS}, alors n_{jS} ajoute n_{iS} à $FAUTIFS_{n_{jS}}$.
   - si la valeur adoptée par n_{jS} est différente de la valeur envoyée par n_{iS}, alors n_{jS} ajoute n_{iS} à $FAUTIFS_{n_{jS}}$.
2. pour chaque nœud n_{iK}, appartenant à un groupe K autre que le groupe source S, il procède comme suit :
   - s'il existe une valeur majoritaire de $V_{n_{iK}}(S)$, n_{iK} adopte cette valeur.
   - s'il n'existe pas une valeur majoritaire de $V_{n_{iK}}(S)$, n_{iK} adopte la valeur de défaut.
3. pour chaque nœud n_{iK} du système, soit $ABSTAIN_{n_{iK}}(T_{KL})$, l'ensemble des nœuds qui n'ont pas envoyé une valeur via le canal T_{KL} (le canal qui connecte le groupe K à un groupe quelconque L). Soit α le cardinal de l'ensemble $ABSTAIN_{n_{iK}}(T_{KL})$.
   - si α est plus grand ou égal à $\left(\frac{\mu-1}{2}\right)$, alors n_{iK} ajoute T_{KL} à $FAUTIFS_{n_{iK}}$, l'ensemble des composants fautifs associés à n_{iK}.

– si α est plus petit que $\left(\frac{\mu-1}{2}\right)$ et si K est le groupe source S, alors n_{iK} ajoute tous les nœuds de l'ensemble $ABSTAIN_{n_{iK}}(T_{KL})$ à $FAUTIFS_{n_{iK}}$, l'ensemble des composants fautifs associés à n_{iK}.

Round 2

begin **diffusion step**

for each node $n_{iS} \in S$, $n_{iS} \neq s$

Broadcast($v_{n_{iS}}$, *);

end for

end **diffusion step**

begin **reception step**

for each node $n_{jS} \in S$, $n_{jS} \neq n_{iS}$

if n_{jS} has received a value from n_{iS} by T_{SK}, $(1 \leq K \leq \gamma \ \& \ K \neq S)$ *then*

$V_{n_{jS}}(n_{iS}) \leftarrow V_{n_{jS}}(n_{iS}) \bigcup \{v_{n_{iS}}\}$;

else

$V_{n_{jS}}(n_{iS}) \leftarrow V_{n_{jS}}(n_{iS}) \bigcup \{\phi\}$;

$ABSTAIN_{n_{jS}}(T_{SK}) = ABSTAIN_{n_{jS}}(T_{SK}) \bigcup \{n_{iS}\}$;

end

end for

for each node $n_{iK} \in K$, $K \neq S$

if n_{iK} has received a value from n_{iS} by T_{SK} *then*

$V_{n_{iK}}(S) \leftarrow V_{n_{iK}}(S) \bigcup \{v_{n_{iS}}\}$;

else

$V_{n_{iK}}(S) \leftarrow V_{n_{iK}}(S) \bigcup \{\phi\}$;

$ABSTAIN_{n_{iK}}(T_{SK}) = ABSTAIN_{n_{iK}}(T_{SK}) \bigcup \{n_{iS}\}$;

end

end for

end **reception step**

begin **decision step**

for each node $n_{jS} \in S$, $n_{jS} \neq n_{iS}$

if ($V_{n_{jS}}(n_{iS})$ is not consistent w.r.t. $v_{n_{iS}}$) *or* ($v_{n_{jS}} \neq$ the value received from n_{iS}) *then*

$FAUTIFSn_{jS} \leftarrow FAUTIFSn_{jS} \bigcup \{n_{iS}\}$;

end

end for

for each node $n_{iK} \in K$, $K \neq S$

$v_{n_{iK}} = maj(V_{n_{iK}}(S))$;

end for

for each node $n_{iK} \in K$, $(1 \leq K \leq \gamma)$

for each group L, $L \neq K$

if $card(ABSTAIN_{n_{iK}}(T_{KL})) \geq \frac{\mu-1}{2}$) *then*

$FAUTIFS_{n_{iK}} = FAUTIFS_{n_{iK}} \bigcup \{T_{KL}\}$;

else if $(card(ABSTAIN_{n_{iK}}(T_{KL})) < \frac{\mu-1}{2}))$ and $(K = S)$

$FAUTIFS_{n_{iK}} = FAUTIFS_{n_{iK}} \bigcup ABSTAIN_{n_{iK}}(T_{KL})$;

end

end for

end for

end **decision step**

TAB. 3.3 – *L'algorithme FIGBA: tour 2.*

3.4.8 Tour 3

Pendant ce tour, chacun des nœuds, n'appartenant pas au groupe source S, diffuse à tous les autres nœuds la valeur qu'il a adoptée si et seulement si cette valeur est différente du ϕ, la valeur nulle. Chaque nœud qui a comme valeur ϕ doit décider quant à la valeur qu'il va adopter en se basant sur les valeurs qu'il va recevoir pendant ce troisième tour. Tous les nœuds sont chargés d'identifier les canaux fautifs. Chaque nœud qui diffuse des valeurs pendant ce troisième tour est chargé d'identifier les nœuds fautifs dans son groupe.

Étape de diffusion : chaque nœud n_{iK}, appartenant au groupe K autre que le groupe source, qui a adopté une valeur différente de ϕ la valeur nulle, se comporte comme une source et diffuse cette valeur. Après cette diffusion, le traitement suivant est effectué :

1. pour chaque nœud n_{jK} [6], appartenant au groupe K,

   - si n_{jK} a reçu une valeur de la source n_{iK}, il ajoute alors cette valeur à $V_{n_{jK}}(n_{iK})$, l'ensemble des valeurs reçues par n_{jK} et envoyées par n_{iK};
   - si n_{jK} n'a pas reçu de valeur via un canal quelconque, disons T_{KL} où L est un groupe donné, alors il ajoute ϕ, la valeur nulle, à $V_{n_{jK}}(n_{iK})$, l'ensemble des valeurs reçues par n_{jK} et envoyées par la source n_{iK}. Il ajoute aussi la source n_{iK} à $ABSTAIN_{n_{jK}}(T_{KL})$, l'ensemble des nœuds qui n'ont pas envoyé une valeur via le canal T_{KL}.

6. Rappelons que lorsqu'un nœud d'un groupe K diffuse une valeur aux autres nœuds via différents canaux, les nœuds de K reçoivent une copie de toutes les valeurs que la source n_{iK} envoie aux nœuds des autres groupes via les différents canaux.

2. pour chaque nœud n_{iL}, appartenant à un groupe dénoté L autre que le groupe source K[7],

   - si n_{iL} a reçu une valeur de la source n_{iK}, il ajoute cette valeur à $V_{n_{iL}}(K)$, l'ensemble des valeurs reçues par n_{iL} et envoyées par les nœuds du K;
   - si n_{iL}, n'a pas reçu de valeur via T_{KL}, alors il ajoute ϕ, la valeur nulle, à $V_{n_{iL}}(K)$, l'ensemble des valeurs reçues par n_{iL} et envoyées par les nœuds du groupe source K. Il ajoute aussi la source n_{iK} à $ABSTAIN_{n_{iL}}(T_{KL})$, l'ensemble des nœuds qui n'ont pas envoyé une valeur via le canal T_{KL}.

Étape de décision : lorsque tous les nœuds, à l'exception des nœuds du groupe source, ont effectué l'étape de diffusion, le traitement suivant est effectué :

1. pour chaque nœud n_{jK}, appartenant au groupe source K, il procède comme suit :

   - si $V_{n_{jK}}(n_{iK})$, l'ensemble des valeurs reçues par n_{jK} et envoyées par le nœud n_{iK}, n'est pas consistant par rapport à la valeur de n_{iK}, alors n_{jK} ajoute n_{iK} à $FAUTIFS_{n_{jK}}$.
   - si la valeur adoptée par n_{jK} est différente de la valeur envoyée par n_{iK}, alors n_{jK} ajoute n_{iK} à $FAUTIFS_{n_{jK}}$.

2. pour chaque nœud n_{iL}, appartenant à un groupe L autre que le groupe source K. Si la valeur adoptée par n_{iL} dans les deux premiers tours est ϕ, la valeur nulle, alors n_{iL} procède comme suit :

   - s'il existe une valeur majoritaire de $V_{n_{iL}}(K)$, n_{iL} adopte cette valeur.
   - s'il n'existe pas une valeur majoritaire de $V_{n_{iL}}(K)$, n_{iL} adopte la valeur de défaut.

7. Rappelons que chaque nœud du groupe L reçoit au plus une valeur de n_{iK} du groupe K via le canal qui connecte son groupe avec le groupe K.

3. pour chaque nœud n_{jK} du système, soit α le cardinal de l'ensemble $ABSTAIN_{n_{jK}}(T_{KL})$, l'ensemble des nœuds qui n'ont pas envoyé une valeur via le canal T_{KL} (le canal qui connecte le groupe K à un groupe L).
   - si α est plus grand ou égal à $\left(\frac{\mu-1}{2}\right)$, alors n_{jK} ajoute T_{KL} à $FAUTIFS_{n_{jK}}$, l'ensemble des composants fautifs associés à n_{jK}.
   - si α est plus petit que $\left(\frac{\mu-1}{2}\right)$ et si K est le groupe source, alors n_{jK} ajoute tous les nœuds de l'ensemble $ABSTAIN_{n_{jK}}(T_{KL})$ à $FAUTIFS_{n_{jK}}$, l'ensemble des composants fautifs associés à n_{jK}.

Round 3

begin **diffusion step**

for each group K, $K \neq S$

for each node $n_{iK} \in K$

if $v_{n_{iK}} \neq \phi$ *then*

Broadcast($v_{n_{iK}}$, *);

end

end for

end for end **diffusion step**

begin **reception step**

for each node $n_{jK} \in K$, $n_{jK} \neq n_{iK}$

if n_{jK} has received a value from n_{iK} by T_{KL}, $(1 \leq L \leq \gamma \ \& \ L \neq K)$ *then*

$V_{n_{jK}}(n_{iK}) \leftarrow V_{n_{jK}}(n_{iK}) \bigcup \{v_{n_{iK}}\}$;

else

$V_{n_{jK}}(n_{iK}) \leftarrow V_{n_{jK}}(n_{iK}) \bigcup \{\phi\}$;

$ABSTAIN_{n_{jK}}(T_{KL}) = ABSTAIN_{n_{jK}}(T_{KL}) \bigcup \{n_{iK}\}$;

end

end for

for each node $n_{iL} \in L$, $L \neq K$, $L \neq S$

if n_{iL} has received a value from n_{iK} by T_{KL} *then*

$V_{n_{iL}}(K) \leftarrow V_{n_{iL}}(K) \bigcup \{v_{n_{iK}}\}$;

else

$V_{n_{iL}}(K) \leftarrow V_{n_{iL}}(K) \bigcup \{\phi\}$;

$ABSTAIN_{n_{iL}}(T_{KL}) = ABSTAIN_{n_{iL}}(T_{KL}) \bigcup \{n_{iK}\}$;

end

end for

end **reception step**

begin **decision step**

for each node $n_{jK} \in K$, $n_{jK} \neq n_{iK}$

if ($V_{n_{jK}}(n_{iK})$ is not consistent w.r.t. $v_{n_{iK}}$) *or* ($v_{n_{jK}} \neq$ the value received from n_{iK}) *then*

$FAUTIFSn_{jK} \leftarrow FAUTIFSn_{jK} \bigcup \{n_{iK}\}$;

end

end for

for each node $n_{iL} \in L$, $L \neq K$

$v_{n_{iL}} = maj(V_{n_{iL}}(K))$;

end for

for each node $n_{iM} \in M$, $(1 \leq M \leq \gamma)$

for each group N, $N \neq M$

if $card(ABSTAIN_{n_{iM}}(T_{MN})) \geq \frac{\mu-1}{2}$) *then*

$FAUTIFS_{n_{iM}} = FAUTIFS_{n_{iM}} \bigcup \{T_{MN}\}$;

else if ($card(ABSTAIN_{n_{iM}}(T_{MN})) < \frac{\mu-1}{2}$)) and $(M = K)$

$FAUTIFS_{n_{iM}} = FAUTIFS_{n_{iM}} \bigcup ABSTAIN_{n_{iM}}(T_{MN})$;

end

end for

end for

end **decision step**

TAB. 3.4 – *L'algorithme FIGBA: tour 3.*

3.4.9 Tour 4

Pendant ce tour, chaque nœud diffuse d'abord à tous les autres nœuds son ensemble des composants fautifs. Chaque nœud va ensuite reconstruire son nouveau ensemble de composants fautifs en se basant sur son ancien ensemble et sur ceux qu'il va recevoir. Finalement, chaque nœud va identifier les canaux fautifs et les nœuds fautifs dans son groupe.

Étape de diffusion : chaque nœud n_{iK} appartenant au groupe K diffuse son ensemble des composants fautifs à tous les autres nœuds. Notons que les nœuds du groupe source K reçoivent une copie de tous les ensembles que la source n_{iK} envoie aux nœuds des autres groupes via les différents canaux. Au contraire, chaque nœud d'un autre groupe reçoit au plus un ensemble de la source n_{iK} via le canal qui connecte son groupe avec le groupe source K. Après cette diffusion, le traitement suivant est effectué :

1. pour chaque nœud n_{jK}, appartenant au groupe K,

   - si n_{jK} a reçu un ensemble de la source n_{iK}, il ajoute alors cet ensemble à $FAUTIFS_{n_{jK}}(n_{iK})$, la liste des ensembles reçus par n_{jK} et envoyés par n_{iK};
   - si n_{jK} n'a pas reçu d'ensemble via un canal quelconque, disons T_{KL} où L est un groupe donné, alors il ajoute $\triangle$, l'ensemble nul, à $FAUTIFS_{n_{jK}}(n_{iK})$, la liste des ensembles reçus par n_{jK} et envoyés par la source n_{iK}. Il ajoute aussi la source n_{iK} à $ABSTAIN_{n_{jK}}(T_{KL})$, l'ensemble des nœuds qui n'ont pas envoyé un ensemble via le canal T_{KL}.

2. pour chaque nœud n_{iL}, appartenant à un groupe dénoté L autre que le groupe source K,

   - si n_{iL} a reçu un ensemble de la source n_{iK}, il ajoute cet ensemble à $FAUTIFS_{n_{iL}}(K)$ la liste des ensembles reçus par n_{iL} et envoyés par les nœuds du K;
   - si n_{iL}, n'a pas reçu d'ensemble via T_{KL}, alors il ajoute $\triangle$, l'ensemble nul, à $FAUTIFS_{n_{iL}}(K)$, la liste des ensembles reçus par n_{iL} et envoyés par les nœuds du groupe source K. Il ajoute aussi la source n_{iL} à $ABSTAIN_{n_{iL}}(T_{KL})$, l'ensemble des nœuds qui n'ont pas envoyé un ensemble via le canal T_{KL}.

Étape de décision : lorsque les étapes de diffusion ont été effectués, nous poursuivons le traitement comme suit :

1. pour chaque nœud n_{jK}, appartenant au groupe source K, il procède comme suit :
   - si $FAUTIFS_{n_{jK}}(n_{iK})$, la liste des ensembles reçus par n_{jK} et envoyés par le nœud n_{iK}, n'est pas consistant par rapport à l'ensemble de n_{iK}[8], alors n_{jK} ajoute n_{iK} à $FAUTIFS_{n_{jK}}$.
   - si l'ensemble adopté par n_{jK} est différent de l'ensemble envoyé par n_{iK}, alors n_{jK} ajoute n_{iK} à $FAUTIFS_{n_{jK}}$.
2. pour chaque nœud n_{iL}, appartenant à un groupe L autre que le groupe source K, il procède comme suit
   - s'il existe un ensemble majoritaire[9] dans la liste des ensembles de fautifs $FAUTIFS_{n_{iL}}(K)$, n_{iL} ajoute cet ensemble à son propre ensemble.

8. on dit que $FAUTIFS_{n_{jK}}(n_{iK})$ est consistant par rapport à l'ensemble de n_{iK} si et seulement si (1) $FAUTIFS_{n_{jK}}(n_{iK})$ consiste en le seul ensemble de n_{iK}; ou (2) $FAUTIFS_{n_{jK}}(n_{iK})$ consiste en deux ensembles : l'ensemble nul $\triangle$ et l'ensemble de n_{iK}.

9. l'ensemble qui se répète le plus souvent

- s'il n'existe pas un ensemble majoritaire dans la liste des ensembles de fautifs $FAUTIFS_{n_{iL}}(K)$, n_{iL} ajoute l'ensemble nul.

3. pour chaque nœud n_{jK} du système, soit $ABSTAIN_{n_{jK}}(T_{KL})$ l'ensemble des nœuds qui n'ont pas envoyé un ensemble via le canal T_{KL} (le canal qui connecte le groupe K à un groupe L). Soit α le cardinal de $ABSTAIN_{n_{jK}}(T_{KL})$.
   - si α est plus grand ou égal à $\left(\frac{\mu-1}{2}\right)$, alors n_{jK} ajoute T_{KL} à $FAUTIFS_{n_{jK}}$, l'ensemble des composants fautifs associés à n_{jK}.
   - si α est plus petit que $\left(\frac{\mu-1}{2}\right)$ et si K est le groupe source, alors n_{jK} ajoute tous les nœuds de l'ensemble $ABSTAIN_{n_{jK}}(T_{KL})$ à $FAUTIFS_{n_{jK}}$, l'ensemble des composants fautifs associés à n_{jK}.

Round 4

begin **diffusion step**

for each group K

for each node $n_{iK} \in K$

Broadcast($FAUTIFS_{n_{iK}}$, *);

end for

end for

end **diffusion step**

begin **reception step**

for each node $n_{jK} \in K$, $n_{jK} \neq n_{iK}$

if n_{jK} has received a set from n_{iK} by T_{KL}, $(1 \leq L \leq \gamma \ \& \ L \neq K)$ *then*

$FAUTIFS_{n_{jK}}(n_{iK}) \leftarrow FAUTIFS_{n_{jK}}(n_{iK}) \bigcup \{FAUTIFS_{n_{iK}}\}$;

else

$FAUTIFS_{n_{jK}}(n_{iK}) \leftarrow FAUTIFS_{n_{jK}}(n_{iK}) \bigcup \{\Delta\}$;

$ABSTAIN_{n_{jK}}(T_{KL}) = ABSTAIN_{n_{jK}}(T_{KL}) \bigcup \{n_{iK}\}$;

end

end for

for each node $n_{iL} \in L$, $L \neq K$

if n_{iL} has received a set from n_{iK} by T_{KL} *then*

$FAUTIFS_{n_{iL}}(K) \leftarrow FAUTIFS_{n_{iL}}(K) \bigcup \{FAUTIFS_{n_{iK}}\}$;

else

$FAUTIFS_{n_{iL}}(K) \leftarrow FAUTIFS_{n_{iL}}(K) \bigcup \{\Delta\}$;

$ABSTAIN_{n_{iL}}(T_{KL}) = ABSTAIN_{n_{iL}}(T_{KL}) \bigcup \{n_{iK}\}$;

end

end for

end **reception step**

begin **decision step**

for each node $n_{jK} \in K$, $n_{jK} \neq n_{iK}$

if ($FAUTIFS_{n_{jK}}(n_{iK})$ is not consistent w.r.t. $FAUTIFS_{n_{iK}}$) *or*

($FAUTIFS_{n_{jK}} \neq$ the set received from n_{iK}) *then*

$FAUTIFSn_{jK} \leftarrow FAUTIFSn_{jK} \bigcup \{n_{iK}\}$;

end

end for

for each node $n_{iL} \in L$, $L \neq K$

$FAUTIFS_{n_{iL}} = FAUTIFS_{n_{iL}} \bigcup maj(FAUTIFS_{n_{iL}}(K))$;

end for

for each node $n_{iM} \in M$, $(1 \leq M \leq \gamma)$

for each group N, $N \neq M$

if $card(ABSTAIN_{n_{iM}}(T_{MN})) \geq \frac{\mu-1}{2}$) *then*

$FAUTIFS_{n_{iM}} = FAUTIFS_{n_{iM}} \bigcup \{T_{MN}\}$;

else if $(card(ABSTAIN_{n_{iM}}(T_{MN})) < \frac{\mu-1}{2}))$ and $(M = K)$

$FAUTIFS_{n_{iM}} = FAUTIFS_{n_{iM}} \bigcup ABSTAIN_{n_{iM}}(T_{MN})$;

end

end for

end for

end **decision step**

TAB. 3.5 – *L'algorithme FIGBA: tour 4.*

3.4.10 Tour 5

Ce tour est en fait identique au quatrième tour. Ainsi, le traitement suivant est effectué : chaque nœud diffuse à tous les autres nœuds l'ensemble des composants fautifs qu'il a adopté au quatrième tour. Chaque nœud va reconstruire son nouveau ensemble de composants fautifs en se basant sur son ancien ensemble et sur ceux qu'il va recevoir. Chaque nœud est chargé d'identifier les canaux fautifs et les nœuds fautifs dans son groupe.

Pour bien comprendre la nécessité de ce tour, examinons le cas suivant : les nœuds d'un groupe K diffusent leurs ensembles des composants fautifs à tous les autres nœuds via les différents canaux connectés au groupe K. Supposons qu'un canal T_{KL} qui connecte le groupe K au groupe L est fautif. Ainsi, les nœuds du groupe L ne reçoivent pas les ensembles de composants fautifs de K. Rappelons que les nœuds du groupe K n'ont pas identifié les nœuds fautifs dans le groupe L. Seuls, les nœuds de L sont chargés d'identifier tous les nœuds fautifs dans leur groupe. Pour cela, ce cinquième tour est nécessaire pour que les nœuds du groupe L puissent recevoir l'ensemble des composants fautifs du groupe K via des canaux non fautifs à partir d'autres groupes.

3.4.11 Illustration

Considérons un système composé de 20 nœuds qui sont répartis sur quatre groupes (G_1, G_2, G_3 et G_4) où $\gamma = 4$ et $\mu = 5$ (figure 3.6). Le nombre des nœuds fautifs tolérés dans chaque groupe est $\frac{\mu-1}{2}$, c'est-à-dire 2 nœuds. Le nombre des canaux fautifs tolérés est $\gamma - 2$, c'est-à-dire 2 canaux. Supposons que les nœuds fautifs sont les nœuds hachurés indiqués par la figure 3.6 (n_{11} et n_{12} dans le groupe G_1, n_{22} et n_{24} dans le groupe G_2, n_{31} et n_{35} dans le groupe G_3 et n_{42} et n_{43} dans le groupe G_4). De même, supposons que les canaux fautifs sont T_{12} et T_{23}.

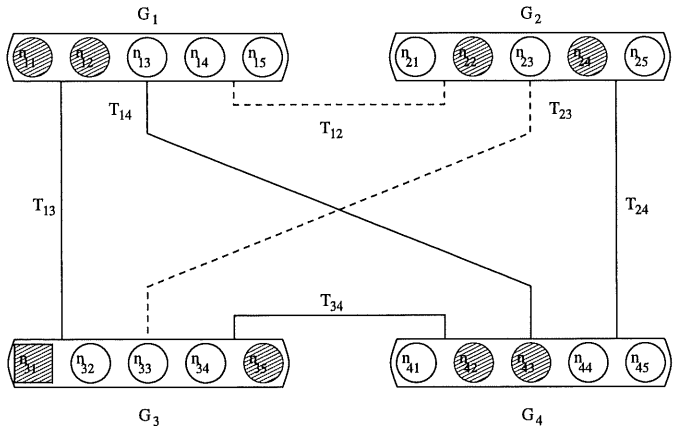


Figure 3.6 – *Un exemple d'un réseau de diffusion généralisé formé de quatre groupes dont chaque groupe contient cinq nœuds .*

Dans ce qui suit, nous présenterons un exemple qui illustre l'algorithme *FIGBA*. Nous traiterons chaque tour séparement. Supposons que n_{31} soit la source. Le groupe G_3 est donc le groupe source. Nous allons décrire la situation du point de vue des nœuds non fautifs. De même, nous allons décrire le comportement des nœuds fautifs. Dans chaque tour, les nœuds passent par trois étapes (étape de diffusion, étape de réception et étape de décision). Nous allons décrire ces trois étapes pour chaque tour.

Tour 1

Ëtape de diffusion : la source diffuse sa valeur à tous les autres nœuds. Supposons que la source n_{31} se comporte comme suit :

- elle s'abstient d'envoyer un message au groupe G_1 via le canal T_{13}.
- elle envoie x au groupe G_2 via le canal fautif T_{23}.
- elle envoie x au groupe G_4 via le canal T_{34}.

La figure 3.7 illustre la situation.

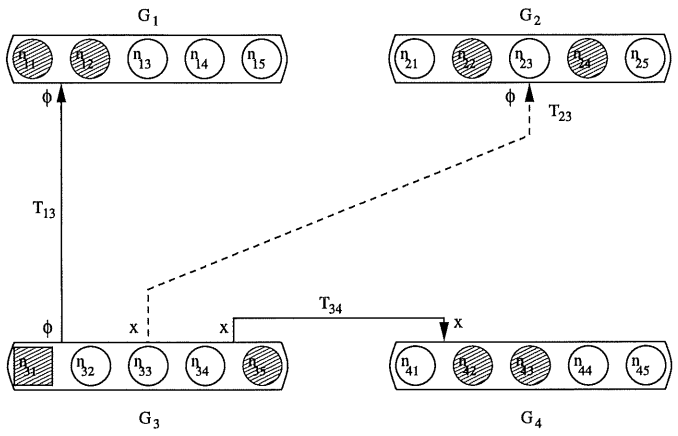


Figure 3.7 – *Diffusion de la source.*

`Étape de réception` : dans chaque groupe, nous allons traiter le cas d'un nœud non fautif. La situation est la même pour tous les nœuds non fautifs dans chaque groupe.

- Traitons le cas d'un nœud non fautif du groupe source G_3, soit n_{32} :
  - l'ensemble des valeurs reçues par n_{32} est :

$$V_{n_{32}}(s) = \{\phi, \phi, x\}$$

  - l'ensemble des nœuds qui n'ont pas envoyé des valeurs via le canal T_{13} est :

$$ABSTAIN_{n_{32}}(T_{13}) = \{n_{31}\}$$

  - l'ensemble des nœuds qui n'ont pas envoyé des valeurs via le canal T_{23} est :

$$ABSTAIN_{n_{32}}(T_{23}) = \{n_{31}\}$$

- Traitons le cas d'un nœud non fautif du groupe G_1, soit n_{13}. La valeur reçue par n_{13} est $v_{n_{13}} = \phi$. Ainsi, $ABSTAIN_{n_{13}}(T_{13})$ est égal au singleton n_{31}.

- Traitons le cas d'un nœud non fautif du groupe G_2, soit n_{21}. La valeur reçue par n_{21} est $v_{n_{21}} = \phi$. Ainsi, $ABSTAIN_{n_{21}}(T_{23})$ est égal au singleton n_{31}.

- Traitons le cas d'un nœud non fautif du groupe G_4, soit n_{41}. La valeur reçue est $v_{n_{41}} = x$.

`Étape de décision` : à la fin de ce tour, tous les nœuds non fautifs du groupe source G_3, adoptent $maj\{\phi, \phi, x\} = x$ (la valeur nulle ϕ n'est pas considéré dans la fonction majorité).

`Remarque` : les nœuds non fautifs, du groupe source G_3, ne peuvent pas savoir pour l'instant que la source est fautive même s'ils n'ont pas reçu de valeur via les deux canaux T_{13} et T_{34}. Pour eux, il existe la possibilité que T_{13} ou T_{23} soient fautifs. Ils vont attendre le second tour pour qu'ils puissent prendre leur décision.

Tour 2

`Étape de diffusion` : tous les nœuds du groupe source G_3, autre que la source n_{31}, diffusent la valeur qu'ils ont adoptée dans le premier tour à tous les autres nœuds. Nous allons traiter le cas d'un nœud non fautif et d'un nœud fautif.

- Un nœud non fautif du groupe source G_3 se comporte comme suit :
  - il envoie x au groupe G_1 via T_{13}.
  - il envoie x au groupe G_2 via le canal fautif T_{23}.
  - il envoie x au groupe G_4 via T_{34}.

  La figure 3.8 illustre la situation.

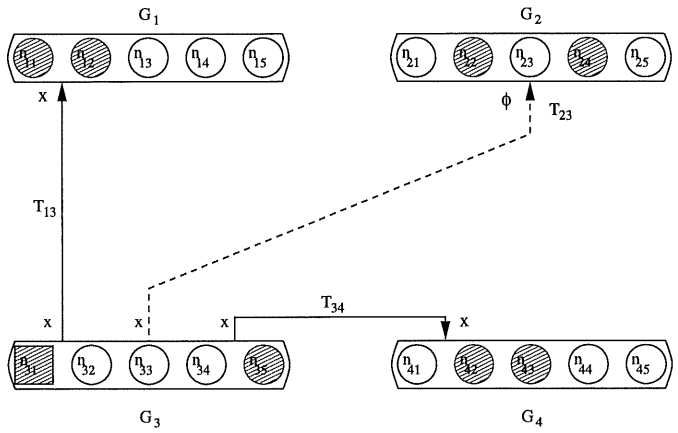


Figure 3.8 – *Diffusion des nœuds du groupe source.*

- Les nœuds fautifs sont libre d'envoyer, de ne pas envoyer ou de changer la valeur adoptée au premier tour. Par exemple, le nœud fautif n_{35} va :
  - s'abstenir d'envoyer un message au groupe G_1 via T_{13}.
  - envoyer x au groupe G_2 via le canal fautif T_{23}.
  - envoyer x au groupe G_4 via T_{34}.

`Étape de réception` : dans chaque groupe, nous allons traiter le cas d'un nœud non fautif après la diffusion des nœuds du groupe source.

- Traitons le cas d'un nœud non fautif du groupe source G_3, soit n_{33}. Le nœud fautif n_{35} n'a pas envoyé de valeur via le canal T_{13}. Ainsi, le nœud n_{33} ajoute n_{35} à l'ensemble des nœuds qui n'ont pas envoyé de message via T_{13}. D'où, $ABSTAIN_{n_{33}}(T_{13}) = \{n_{31}, n_{35}\}$. De même, aucune valeur n'est passée via le canal fautif T_{23}. Ainsi, le nœud n_{33} ajoute tous les autres nœuds du groupe source à l'ensemble des nœuds qui n'ont pas envoyé de message via T_{23}. D'où, $ABSTAIN_{n_{33}}(T_{23}) = \{n_{31}, n_{32}, n_{34}, n_{35}\}$.

- Traitons le cas d'un nœud non fautif du groupe G_1, soit n_{13}. L'ensemble des valeurs reçues par n_{13} est $V_{n_{13}}(G_3) = \{\phi, x, x, x, \phi\}$. Le nœud n_{13} n'a pas reçu de valeur de n_{31} et de n_{35} via le canal T_{13}. Ainsi, $ABSTAIN_{n_{13}}(T_{13}) = \{n_{31}, n_{35}\}$.
- Traitons le cas d'un nœud non fautif du groupe G_2, soit n_{21}. L'ensemble des valeurs reçues par n_{21} est $V_{n_{21}}(G_3) = \{\phi, \phi, \phi, \phi, \phi\}$. Le nœud n_{21} n'a pas reçu aucune valeur via le canal fautif T_{23}. Ainsi, $ABSTAIN_{n_{21}}(T_{23}) = \{n_{31}, n_{32}, n_{33}, n_{34}, n_{35}\}$.
- Traitons le cas d'un nœud non fautif du groupe G_4, soit n_{41}. L'ensemble des valeurs reçues par n_{14} est $V_{n_{41}}(G_3) = \{x, x, x, x, x\}$.

`Étape de décision :` à la fin de ce tour, tous les nœuds non fautifs dans tous les groupes (autre que le groupe source) se comporte de la façon suivante :

1. chaque nœud non fautif du groupe G_1 adopte $maj(\phi, x, x, x, \phi) = x$.
2. les nœuds non fautifs du groupe G_2 n'adoptent rien puisqu'ils n'ont rien reçu.
3. chaque nœud non fautif du groupe G_4 adopte $maj(x, x, x, x, x) = x$

À la fin de ce tour, nous avons aussi ce qui suit. Tous les nœuds non fautifs dans tous les groupes vont identifier, à partir des informations disponibles jusqu'ici, les nœuds et les canaux fautifs. Plus précisement, on a :

1. pour un nœud non fautif du groupe G_1, (par exemple n_{13}), l'action entreprise est :

   $ABSTAIN_{n_{13}}(T_{13}) = \{n_{31}, n_{35}\}$ alors n_{13} ajoute n_{31} et n_{35} à $FAUTIFS_{n_{13}}$

2. pour un nœud non fautif du groupe G_2, (par exemple n_{21}), l'action entreprise est :

   $ABSTAIN_{n_{21}}(T_{23}) = \{n_{31}, n_{32}, n_{33}, n_{34}, n_{35}\}$ alors n_{21} ajoute T_{23} à $FAUTIFS_{n_{13}}$

3. pour un nœud non fautif du groupe source G_3, (par exemple n_{32}), l'action entreprise est :

$$ABSTAIN_{n_{32}}(T_{13}) = \{n_{31}, n_{35}\} \text{ et } ABSTAIN_{n_{32}}(T_{23}) = \{n_{31}, n_{32}, n_{33}, n_{34}, n_{35}\}$$

alors n_{32} ajoute n_{31}, n_{35}, et T_{23} à $FAUTIFS_{n_{32}}$

4. tous les nœud non fautifs du groupe G_4 ne peuvent rien identifier.

Tour 3

Chaque nœud qui n'appartient pas au groupe source G_3 et qui a déjà adopté une valeur au second tour, diffuse sa valeur. Dans notre cas, seul les nœuds du groupe G_1 et du groupe G_4 vont diffuser de valeurs. Par exemple un nœud non fautif du groupe G_4, soit n_{41}, envoie x au groupe G_1 via T_{14} et x au groupe G_2 via T_{24}. La figure 3.9 illustre la situation.

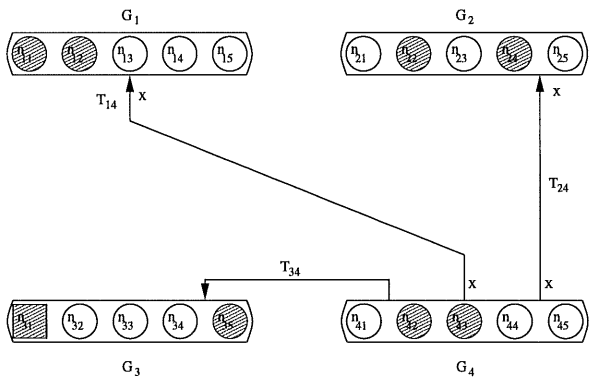


Figure 3.9 – *Diffusion des nœuds du groupe G_4.*

À la fin de ce tour, le seul groupe qui n'a pas encore adopté une valeur, c'est le groupe G_2. Chaque nœud non fautif du groupe G_2 reçoit une majorité de la valeur x. Il va ainsi

adopter cette valeur. Tous les nœuds non fautifs dans tous les groupes vont identifier, à partir des informations disponibles jusqu'ici, les nœuds et les canaux fautifs.

Tour 4 et 5

Dans le quatrième tour, chaque nœud n_{iK}, dans un groupe K, diffuse $FAUTIFS_{n_{iK}}$ à tous les autres nœuds. Chaque nœud n_{jL} collecte tous les ensembles envoyés par un groupe K dans un ensemble, dénoté $FAUTIFS_{n_{jL}}(K)$. À la fin de ce tour, n_{jL} ajoute $maj\left(FAUTIFS_{n_{jL}}(K)\right)$ à son ensemble des composants fautifs.

Dans le cinquième tour, chaque nœud qui a déduit un nouvel ensemble à la fin du quatrième tour, diffuse cet ensemble. Le même scénario du quatrième tour se répète.

3.4.12 Conclusion

Dans ce chapitre, nous avons développé un algorithme qui résout le BA et qui identifie tous les composants fautifs dans le système (nœuds et lignes de communication). L'architecture de réseau utilisé est efficace. Elle répartit les nœuds dans γ groupes dont chaque groupe contient μ nœuds. Dans un système composé de $\mu\gamma$ nœuds, le nombre des nœuds fautifs toléré est $\frac{\mu-1}{2}\gamma$ par comparaison avec $\frac{\mu\gamma}{3}$ dans la solution traditionnelle du BA, [13]. Le nombre de tours requis est cinq par comparaison avec f_n+1 dans la solution traditionnelle du BA, [13].

Conclusion

Au cours de ce travail, nous avons conçu une méthode d'identification des composants fautifs pour le problème des généraux Byzantins. Deux objectifs ont été atteints :

1. À partir du problème Byzantin, nous avons conçu un mécanisme d'identification des composants fautifs. Ce mécanisme d'identification n'a rien ajouté au coût de la résolution du *BA*, que ce soit au niveau des communications ou au niveau du nombre de tours. Nous avons aussi montré qu'il existe des cas où il est impossible d'identifier tous les composants fautifs.

2. Nous avons développé un algorithme, dans une architecture de communication combinant le réseau connecté complet et le réseau de diffusion, qui respecte l'accord Byzantin et qui identifie tous les composants fautifs dans le système. Ainsi, les pannes des nœuds et des lignes de communication ont été considérées.

Ces deux points constituent les principales contributions de notre mémoire.

Des travaux futurs de développement pourraient se résumer comme suit :

1. Énoncer un nouvel ensemble de conditions nécessaires et suffisantes pour l'identification des composants fautifs,

2. Mettre au point les algorithmes d'identification des composants fautifs,

3. Analyser les performances des algorithmes d'identification proposés.

Bibliographie

[1] O. Babaoglu and R. Drummond. Streets of byzantium: Network architectutres for fast reliable broadcasts. *IEEE Transactions on Software Engineering*, SE-11, 1985.

[2] O. Babaoglu, P. Stephenson, and R. Drummond. Reliable broadcasts and communication models: Tradeoffs and lower bounds. *Dist. Computing*, 2, 1988.

[3] T. Bracha and S. Toueg. Asynchronous consensus and broadcast protocols. *Journal of the ACM*, 32(4), 1985.

[4] Ran Canetti and Tal Rabin. Fast asynchronous byzantine agreement with optimal resilience. *25th ACM STOC*, 5(42), 1985.

[5] Danny Dolev. The byzantine generals strike again. *Journal of algorithms*, 3, 1982.

[6] B. El-Ayeb. Fiabilité des systèmes. In *Cours IFT743*. Université de Sherbrooke, 1995.

[7] M. Fischer, N. Lynch, and M. Paterson. Impossibility of distributed consensus with one faulty processor. *Journal of the Association for Computing Machinery*, 32(2), 1985.

[8] Ajei Gopal and Sam Toueg. Reliable broadcast in synchronous and asynchronous environments. Technical Report 14853, Departement of Computer Science, Upson Hall, Cornell University, 1992.

[9] R.M. Kieckhafer and M.H. Azadmanesh. Reaching approximate agreement with mixed-mode faults. *IEEE Transactions on Parallel and Distributed Systems*, 5(1), january 1994.

[10] Thijs Krol. Interactive consistency algorithms based on voting and error-correcting codes. *IEEE*, 1995.

[11] L. Lamport. The weak byzantine generals problem. *Journal of the Association for Computing Machinery*, 30(3), july 1983.

[12] L. Lamport and P. M. Melliar-Smith. Synchronizing clocks in the presence of faults. *ACM Transactions on Programming Languages & Systems*, 4(3), 1982.

[13] L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem. *Journal of the Association for Computing Machinery*, 32(1), january 1985.

[14] M. Molcho and S. Zaks. Robust asynchronous algorithms in networks with a fault detection ring. *LNCS*, 1994.

[15] J. Pankaj. *Fault Tolerance in Distributed Systems.* Department of Computer Science and Engineering Indian Institute of Technology, Kanpur, 1994.

[16] M. Pease, R. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *Journal of the ACM*, 27(2), 1980.

[17] F. Preparata, G. Metze, and R. Chien. On the connection assignment problem of diagnosable systems. *IEEE Trans. Elect. Comput.*, EC-16(6), 1967.

[18] W. Stalling. *Data and Computer Communications.* New York: Macmillan, 1985.

[19] W. Stallings. *Data and Computer Communications.* Prentice Hall, Upper Saddle River, New Jersey 07458, 1996.

[20] N.H. Vaidya and D.K. Pradhan. Degradeable agreement in the presence of byzantine faults. In *Proceedings the 13th International Conference on Distributed Computing Systems IEEE Computer Society*, 1993.

[21] S.C. Wang, Y.H. Chin, and C. Chen. Achieving byzantine agreement in a generalized connected network model. In *Proceedings CompEuro 89, Hamburg, Germany*, 1989.

[22] S.C. Wang, Y.H. Chin, and K.Q. Yan. Byzantine agreement in a generalized connected network. *IEEE Transactions on Parallel and Distributed Systems*, 6(4), 1995.

[23] K.Q. Yan and Y.H. Chin. Achieving byzantine agreement in a processor & link fallible network. In *Proceedings International Phoenix Conference Comput., communicat.*, 1989.

[24] Kuo-Qin Yan, Y.H. Chin, and Shu-Ching Wang. Optimal agreement protocol in malicious faulty processors and faulty links. *IEEE Transactions on Knowledge and Data Engineering*, 4(3), june 1992.